Universidad
Carlos III de Madrid

DEPARTAMENTO DE CIENCIA DE LA
COMPUTACIÓN E INTELIGENCIA ARTIFICIAL

# Diseño y desarrollo de un jugador inteligente para la competición anual de agentes de póquer

---

## PROYECTO FIN DE CARRERA

**Autor: Bruno Martín Gayral**
**Director: Agapito Ledezma Espino**

**Leganés, Julio 2012**

Título: Diseño y desarrollo de un jugador inteligente para la competición anual de agentes de póquer
Autor:
Director:

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

# Agradecimientos

Este trabajo no hubiese sido posible realizarlo sin la ayuda y el apoyo que he recibido de mucha gente:

Gracias a mi familia y sobretodo de mis padres Antonio y Annie por su infinita paciencia y apoyo.

Gracias a mi tutor en el proyecto y también profesor Agapito Ledezma, por su apoyo, paciencia y motivación en los momentos difíciles.

Gracias a Michael Bowling y al equipo de desarrollo de agentes de póquer de la Universidad de Alberta, por su buena disposición y ayuda para realizar las pruebas de rendimiento frente su agente campeón del mundo. También por ofrecer su servidor e interfaz de juego gratuitamente, que ha sido de mucha ayuda durante el desarrollo y la realización de las pruebas.

Gracias a todos mis profesores de carrera por los conocimientos que me transmitieron en su día y que en definitiva me han permitido adquirir las habilidades para llevar a cabo este tipo de proyectos.

Gracias a los creadores del proyecto Pokersource por su librería de evaluación de manos de póquer de código abierto, que permite que Uc3mBot identifique la fuerza de su mano en cada una de las distintas situaciones del juego a una gran velocidad.

Gracias a James Devlin por sus artículos acerca del desarrollo de jugadores de póquer por internet y más especialmente por su librería, XPokerEval la cual me facilitó enormemente el trabajo a la hora de realizar simulaciones de manos.

Gracias a la comunidad de desarrolladores de poker.ai, por su resolución de dudas y por toda la información que ofrecen.

Gracias a Andrew Prock por su calculadora de equidades “Pokerstove” que me ha permitido verificar la precisión de los cálculos del agente creado en este PFC.

# Resumen

La Inteligencia Artificial (IA) ha sido criticada por centrarse en la resolución de problemas de dominio determinista e información perfecta, como en el juego de ajedrez o las damas, en los que se puede desarrollar un agente de alto nivel de juego aplicando el algoritmo de búsqueda Minimax. Sin embargo muchos problemas de la vida real manifiestan propiedades no deterministas y de información incompleta.

El juego del póquer, que presenta este tipo de dominio, es un escenario perfecto para crear nuevas técnicas y avanzar en el campo de la IA. El objetivo de este proyecto ha sido desarrollar un agente de póquer que juegue en la modalidad Texas Hold'em sin límite, para partidas de dos jugadores, a un nivel adecuado para presentarlo a la Competición Anual de Agentes de Póquer (ACPC), la competición más importante hasta la fecha de este tipo de agentes. Entre otros requisitos importantes se ha querido poner especial énfasis en que el agente pudiese modelar el comportamiento del oponente, de manera que el programa fuese adaptando su juego al estilo del adversario. Los resultados obtenidos demuestran que la estrategia adoptada está a la altura de dicha competición.

# Contenido

# Índice de Figuras

# Índice de Tablas

# Índice de Gráficos

# Capítulo 1

## Introducción y objetivos

Los juegos, que siempre han supuesto un reto para la mente humana, han demostrado ser una excelente herramienta de investigación para la informática y en concreto para la Inteligencia Artificial (IA). Esto es debido a que la mayoría comparte una lógica sencilla frente a la necesidad de desarrollar una estrategia compleja, pues todos tienen una serie de reglas bien definidas, un objetivo específico que cumplir, y gracias a la gran cantidad de expertos existentes, proporcionan una forma relativamente fácil de medir los progresos.

Desde que Alan Turing en 1951 crease en papel el primer autómata capaz de jugar una partida completa de ajedrez, los programadores de computadoras han desarrollado varios ejemplos de jugadores por ordenador capaces de batir a los mejores humanos. Tres de los más importantes son:

- Logistello. En 1997 este programa, creado por Michael Buro, ganó todas y cada una de las 6 partidas contra el por entonces campeón del mundo Takeshi Murakami, dejando pocas dudas al respecto de la superioridad de los programas frente a los humanos en el juego del Othello.
- Chinook. Creado por Jonathan Schaeffer y sus colaboradores, fue el primer programa en ganar un título de campeón del mundo en el juego de las damas. Ocurrió durante el campeonato hombre-máquina (Man-Machine World

Championship) de 1994, donde se batió contra el campeón de entonces Marion Tinsley. En Julio de 2007 los desarrolladores de Chinook anunciaron que el programa había sido mejorado hasta el punto en que es imposible que pierda.

- Deep Blue. El ordenador que batió a Kasparov en mayo de 1997, ganando el encuentro a 6 partidas por 3½-2½, lo que lo convirtió en la primera computadora en derrotar a un campeón del mundo vigente, en un encuentro con ritmo de juego de torneo estándar.

Por ende se puede afirmar que hoy en día los juegos continúan siendo plataformas altamente atractivas para el desarrollo científico.

## 1.1. Objetivos de este Proyecto Fin de Carrera

Para el desarrollo de este PFC se ha tenido como fin cumplir los siguientes objetivos:

1) Desarrollar un agente de póquer Texas Hold'em sin límite en la modalidad de 2 jugadores, que despliegue un juego inteligente, capaz de:
   a) Adaptarse a la forma de jugar del oponente y desarrollar contraestrategias en tiempo de ejecución que le permitan maximizar las ganancias en el juego.
   b) Determinar la fuerza relativa de su mano en cualquier situación de juego.
   c) Estimar la probabilidad de ganar en cada una de las situaciones de juego.
   d) Desarrollar un juego poco predecible, que no actúe siempre igual ante la misma situación de juego, y que sea capaz de tirarse faroles.
2) El programa ha de poder comunicarse con el servidor de juego proporcionado por la Universidad de Alberta, para en un futuro si se desea, poder presentar el agente a la Annual Computer Poker Competition.
3) El agente desarrollado ha de ser fácil de adaptar en el futuro a otros servidores del juego.
4) Incluir una interfaz gráfica que permita jugar contra el programa y probar su nivel de juego frente a jugadores humanos.

5) La toma de decisiones del programa no debe llevar un tiempo excesivo. El juego debe ser lo más fluido posible para poder efectuar pruebas frente a distintos oponentes humanos y obtener resultados en un tiempo razonable.

## 1.2. ¿Por qué un agente de póquer?

La IA ha obtenido grandes éxitos y avances a través de juegos típicamente deterministas (sin incertidumbre) y de información perfecta como el ajedrez o las damas; juegos donde todos los jugadores pueden determinar la situación exacta del juego y no interviene el azar. Es por ello que otros tipos de juegos, como los de información imperfecta (donde parte de la información permanece oculta a los jugadores), abren nuevas posibilidades a avances en este campo.

| | **Deterministas (sin incertidumbre)** | **No deterministas (interviene el azar)** |
|---|---|---|
| **Información perfecta (no hay información oculta)** | Ajedrez, damas, go | Backgammon, Parchís, La oca |
| **Información imperfecta (hay información oculta)** | Hundir la flota, piedra papel o tijera | Póquer, Mus, Bridge |

Tabla 1: Tipos de juegos

El póquer supone un desafío diferente a otros juegos estudiados, pues además de cumplir estos requisitos presenta otras características muy interesantes, como son:

- **La necesidad de modelar el comportamiento de los oponentes**. No existe un estilo de juego que sea el más efectivo para todos los casos, el póquer es un juego en que es muy importante identificar la manera en que juega el oponente, determinando su nivel de agresividad o si es un contrincante selectivo o no. Todos estos factores son fundamentales a la hora de enfrentarse en la mesa.

- **Adaptar el estilo de juego en función del tipo de contrincante**. Una vez determinado qué forma de juego está desarrollando el contrario, es preciso adaptar nuestro juego al suyo de manera que maximicemos las ganancias. Por ejemplo, si el adversario juega de un modo poco selectivo y pasivo nosotros buscaremos un estilo selectivo y agresivo a su vez, pues a la larga jugaremos con un rango de cartas mejor que el del oponente y nuestra mayor agresividad le hará abandonar manos ganadoras con más frecuencia, lo que supone una gran ventaja.

- **Enfrentar resultados estocásticos**. En el póquer no se sabe con certeza qué cartas se van a repartir, de modo que es imprescindible trabajar en términos de probabilidades. Es más, en el póquer no es raro haber jugado una mano de manera perfecta y aun así perder debido a la mala suerte. Por esto es importante hablar de maximizar las ganancias a largo plazo y determinar cuánto estamos ganando de media cada vez que realizamos una jugada u otra, en vez de centrarnos en cuánto hemos ganado o perdido en una mano concreta.

- **La necesidad de introducir impredecibilidad y engaño en el juego**. Un jugador deja de ser bueno en cuanto su juego se hace predecible, pues a partir de ése momento resultará muy fácil para el contrincante determinar la fuerza de sus cartas, y por lo tanto le será igual de fácil sacar provecho de ello. Por este motivo es imprescindible no jugar siempre los mismos tipos de cartas de la misma manera y tratar de engañar al oponente, haciéndole pensar que llevamos unas cartas distintas a las nuestras. De este modo un buen jugador irá cambiando su estilo de juego durante el transcurso de una partida para que las estrategias adoptadas frente a él dejen de ser efectivas. Así es cómo este juego se convierte en una continua adaptación al adversario.

- **Lidiar con información poco fiable**. Cada jugador sólo conoce sus cartas y las cartas comunitarias[1], todo lo demás son suposiciones sobre el juego puesto que

[1] Cartas que se reparten boca arriba en la mesa y que cualquier jugador podría usar. Pertenecen a todos los jugadores, por eso son comunes. En Texas Hold'em, por ejemplo, las tres cartas del flop, la carta del turn y la carta del river son las cartas comunitarias.

existe un número enorme de posibles manos que el oponente puede tener. Parte del desafío del póquer consiste en adivinar qué cartas podría llevar el oponente dado sus acciones en la mano actual, su estilo de juego, el tamaño del bote, las posibles jugadas con proyectos, etc.

- **Competir contra múltiples agentes.** El póquer puede jugarse en partidas de 2 hasta 10 jugadores, aunque en la práctica podrían ser hasta 23 (en la versión Texas Hold'em). El hecho de poder competir contra más de 1 jugador añade nuevos retos al juego.

## 1.3. ¿Por qué la variante Hold'em sin límite?

La palabra "póquer" es el nombre genérico de centenares de juegos distintos, aunque pueden agruparse en unas cuantas variedades relacionadas. Hay juegos *high* donde gana la mano más alta a la hora de enseñar, como son el Texas Hold'em y el Seven Card Stud; y juegos *low* en los que gana la mano más baja, como el Razz y el Triple-draw. Incluso hay juegos *high-low* en los que se reparten el bote la mano más alta y la más baja. Las variantes del póquer también se pueden clasificar por estructura de apuestas. Existen los juegos con límite de apuestas (*limit*), otros donde la apuesta máxima es el tamaño del bote (*pot-limit*), y otros donde no hay límite de apuestas (*no-limit*).

Se ha elegido estudiar el juego del Texas Hold'em en su versión sin límite de apuestas por las siguientes razones:

- Existe mucho terreno avanzado en el Hold'em con límite, hasta el punto que se han desarrollado agentes capaces de batir a la mayoría de oponentes humanos (1). Sin embargo esto no ocurre en la variante sin límite, debido principalmente a que el número de posibles jugadas a efectuar en cada decisión de juego es mayor, y por tanto también su complejidad estratégica, lo que convierte a este

PFC en un reto más interesante, pues a pesar de que se han realizado algunos avances, queda aún mucho camino por recorrer.

- Es la variante del póquer considerada como **más compleja estratégicamente** por la mayoría de jugadores, sobre todo cuando se juega con una pila grande de fichas. Además está considerada como la modalidad donde más prima la habilidad sobre la suerte.
- Existen más facilidades a la hora de desarrollar programas para el Texas Hold'em que para cualquier otra variante. Ejemplo de ello son la buena cantidad de librerías de código abierto que se pueden utilizar para facilitar cálculos de mano ganadora y simulaciones.
- Las reglas básicas del Texas Hold'em son más sencillas que las de otras variantes de póquer, y por lo tanto su implementación.
- Es la modalidad de póquer más jugada hoy en día, lo que facilita el acceso a una mayor cantidad de expertos en el juego. La variante Hold'em sin límite es además la utilizada para determinar cada año al campeón del mundo en el evento principal de las Series Mundiales de Póquer (WSOP[2]).

## 1.4. Distribución del documento

Tras ésta introducción al proyecto, y una vez explicados punto por punto los objetivos perseguidos, explicando qué se ha buscado hacer y qué características se pretendía que tuviese el jugador de póquer por ordenador. Éste es un resumen de lo que nos aguarda en los siguientes capítulos:

En el capítulo 2 se detallará el estado de la cuestión o estado del arte, en él se describen los avances más importantes realizados hasta la fecha en el desarrollo de agentes de póquer y los principales artífices de estos. Incluye descripciones, modelos e ideas que se han utilizado para desarrollar el proyecto.

El capítulo 3 es el más extenso puesto que es la memoria del trabajo realizado. Comenzando por una descripción de alto nivel hasta los detalles más importantes.

[2] del ingles World Series of Poker

Incluye una explicación detallada de la arquitectura describiendo de las diferentes partes o módulos, el diagrama de clases, el manual de usuario para ejecutar la aplicación, con una descripción de la interfaz gráfica y los pasos a seguir, y finalmente el manual de referencia para futuras ampliaciones del software. Dicho manual contiene los detalles técnicos de implementación, el lenguaje utilizado, las librerías de cálculo, los ficheros generados, etc.

El capítulo 4 expone todas las pruebas realizadas para probar el correcto funcionamiento y el cumplimiento de los objetivos propuestos al principio de este PFC, así como la interpretación de los resultados obtenidos.

Finalmente el capítulo 5 incluye las conclusiones y futuras líneas de trabajo respectivamente.

# Capítulo 2

## Estado del arte

No deja de sorprender el hecho de que pese a que el póquer se ha utilizado mucho como tema de estudio en las matemáticas o la estadística, la comunidad informática no le prestó demasiada atención hasta hace relativamente pocos años. Por ejemplo matemáticos de la talla de John von Neumann (2) y John Nash (3) usaron versiones simplificadas de póquer para explicar principios fundamentales de la teoría de juegos a principios de los años 50.

Este capítulo busca acercarse a los avances realizados en el estudio del póquer desde dos perspectivas distintas y siguiendo un orden cronológico. En primer lugar se describen los trabajos que se acercan al póquer desde una perspectiva teórica, principalmente utilizando versiones simplificadas del juego, primero para explicar principios de la teoría de juegos y luego para estudiar las dinámicas del póquer en sí. Los siguientes apartados tratan el juego desde un punto de vista más práctico, aplicado a un póquer más real. En ellos se repasan los avances más importantes realizados en el campo de la IA y la creación de jugadores de póquer por computador.

## 2.1. La teoría de juegos: Un acercamiento teórico

Los primeros estudios académicos relacionados con el póquer se llevaron a cabo dentro de la rama de las matemáticas. En el extenso libro "Theory of Games and Economic Behavior" (2), el trabajo que formalizó la teoría de juegos, John von Neumann y Oskar Morgenstern utilizaron modelos matemáticos para analizar una versión simplificada del póquer en la modalidad Stud[3]. A parte de utilizar el póquer para explicar los principios de dicha teoría, distinguieron dos formas de ir de farol[4], una puramente agresiva y otra más defensiva, y demostraron que este componente es absolutamente esencial en el juego, afirmando que cualquier estrategia que se precie debe incluir ir de farol con una frecuencia determinada.

A pesar de lo interesante y útil de este estudio la versión que se empleó del juego fue extremadamente simple, ya que se limitaba el número de apuestas excluyendo la posibilidad de hacer resubidas, situándolo lejos de los juegos de póquer reales. En palabras de Neumann: *"El póquer actual es realmente un tema demasiado complicado para una discusión exhaustiva y por lo tanto tendremos que someterlo a simplificación mediante algunas modificaciones, algunas de las cuales son, de hecho, bastante radicales".*

Continuando con el estudio de versiones simplificadas de póquer llegamos al trabajo desarrollado por John Nash, quien recuperó las ideas de Neumann en su brillante tesis "Non-cooperative games" (4), de menos de treinta páginas, para demostrar su concepto de *punto de equilibrio* [5] utilizando una versión simplificada de póquer de 3 jugadores. En dicho trabajo explica el procedimiento para encontrar todos los puntos de equilibrio del juego y concluye que un análisis de una versión más compleja de póquer podría ser muy interesante, aunque admite que la complejidad del estudio

[3] Modalidad de póquer en las cuales el jugador recibe un cierto número de cartas, alguna de ellas boca abajo y otras boca arriba. También conocido como póquer descubierto.
[4] Hacer uso del engaño apostando sin tener una mano ganadora, con intención de que el adversario abandone.
[5] Situación en la cual todos los jugadores han puesto en práctica, a sabiendas, una estrategia que maximiza sus ganancias dadas las estrategias de los otros.

aumentaría considerablemente, de modo que sólo sería factible estudiar versiones de póquer real mediante cálculos por ordenador y métodos aproximados.

Poco desarrollo más ha habido en la línea de estos primeros estudios, probablemente debido a que fueron creados como una lección en el uso de la teoría de juegos en vez de tratarse de una investigación de las dinámicas del póquer en sí. Consecuentemente, estos modelos tienen poco uso práctico a la hora de escribir un algoritmo[6] que juegue un póquer sólido. Sin embargo, las técnicas utilizadas para determinar la validez o efectividad de estrategias de juego como el farol (y la frecuencia idónea de su uso), son dos ideas útiles que se pueden extraer de la teoría de juegos.

Más adelante William Cutler (5) desarrolló en papel una estrategia óptima para una variante del póquer tapado, con apuestas limitadas al tamaño del bote (pot-limit) para partidas de dos jugadores. Como los trabajos anteriores a éste, dicho análisis se basaba en una forma simplificada de póquer, en este caso con una única ronda de apuestas y sin la posibilidad de pedir cartas.

La estrategia obtenida resultó ser bastante rígida ya que siempre jugaba del mismo modo con cartas de igual valor, es decir, el número de apuestas era proporcional al valor de las cartas, lo que invariablemente no dejaba lugar al engaño y desplegaba un modo de juego muy predecible. Tampoco incluía forma alguna de adaptar el juego al estilo del oponente, lo que hacía a esta estrategia aún más vulnerable. El propio Cutler afirma en sus conclusiones que se trata de una estrategia muy conservadora y no se obtienen por lo tanto demasiados beneficios con ella. A pesar de ello, aportó un método de análisis que puede generalizarse para estudiar partidas con cualquier número de subidas, lo cual se acerca un poco más a un juego de póquer real que los estudios anteriores, donde era posible efectuar únicamente una subida o ninguna.

Acorde con la línea anterior Norman Zadeh (6) publicaba en 1974 el volumen "Winning Poker Systems" donde analiza meticulosamente numerosas variables del póquer tapado de cinco cartas. El libro está lleno de gráficos que determinan las manos con las que es correcto jugar, para lo que tiene en cuenta conceptos del juego

[6] Conjunto de instrucciones o reglas bien definidas, ordenadas y finitas que permite realizar una actividad mediante pasos sucesivos que no generen dudas

que hoy en día resultan básicos; como la posición, las posibilidades matemáticas de ligar los distintos proyectos, las probabilidades del bote, el número de jugadores y el tamaño de las ciegas. También examina otras modalidades de póquer como son el Lowball, el Stud y variantes High-Low. Al final con las lecciones que obtiene del estudio de cada variante elabora una serie de reglas generales que pueden aplicarse a todas las variantes del póquer.

En 1981, la publicación “Poker Strategy: Winning with Game Theory” (7) hace un ambicioso análisis teórico de un juego de póquer real. En él Nesmith Ankeny, ingenia a través de la teoría de juegos una estrategia cuasi óptima para jugar al póquer tapado de cinco cartas, la variante más popular en aquellos años. Dicha estrategia consiste en dividir el juego por fases para analizarlo como problemas separados y así reducir la complejidad de su estudio.

Las soluciones se presentan en forma de una serie de tablas que contienen los distintos tipos de manos posibles para cada fase, junto con las reglas que determinan cuando apostar, igualar o pasar, en función del tipo de mano.

A pesar de que es probable que la estrategia presentada reporte cierto margen de beneficios incluso frente a oponentes sólidos (debido a su tendencia más conservadora), el hecho de estudiar cada fase de una mano de forma aislada supone no tener en cuenta mucha información valiosa; ya que el póquer es un juego en el que influyen mucho el contexto y las acciones pasadas. Además se trata de nuevo de una estrategia que desarrolla un juego predecible y no saca ventaja de oponentes más débiles.

## 2.2. Los estudios de Findler: Primeras aplicaciones informáticas

Los primeros trabajos de la informática centrados en el póquer fueron llevados a cabo por el profesor Nicolas Findler y sus colaboradores entre 1961 y 1980. En esos trabajos se utilizó de nuevo la versión del póquer tapado de cinco cartas como modelo para una variedad de estudios dentro de la IA (8) (9) (10) (11) (12) (13). El objetivo de dichos estudios era simular los procesos de pensamiento de jugadores humanos, desde los de un jugador principiante a un jugador experto.

En su primer artículo publicado (8) Findler elabora un informe sobre la posibilidad de desarrollar un algoritmo que permitiese a un ordenador jugar al póquer. Para lo que detalla el procedimiento en un organigrama y aporta un algoritmo en pseudocódigo que resulta algo incompleto, pues sólo tiene en cuenta el caso en que el agente juega último; y que desarrolla un juego predecible, ya que determina la cantidad de la apuesta en proporción directa con la fuerza de la mano, lo que para un jugador inteligente representa una gran ventaja, pues le permite estimar más fácilmente las cartas del oponente.

A pesar de esto Findler fue uno de los primeros investigadores en aplicar técnicas de aprendizaje computacional en juegos, representando un modo distinto de acercarse al problema y dejando a un lado la teoría de juegos, ya que pensaba que un enfoque puramente matemático no resultaría fructífero dada la imposibilidad de aplicarla a un juego de póquer real.

Desafortunadamente, las estrategias que utilizó para explorar este campo podrían clasificarse en la actualidad como de arbitrarias dada la baja utilidad de los resultados que obtuvo. Aun así es importante subrayar que Findler enfocó sus esfuerzos en simular la forma de juego de los humanos y no en desarrollar el mejor programa de póquer.

## 2.3. El CPRG: Creando un campeón de póquer por ordenador

No fue hasta 1995 que el grupo de investigadores del CPRG (Computer Poker Research Group) de la Universidad de Alberta (Edmonton, Canadá), con amplia experiencia en el desarrollo de jugadores por ordenador (14), comenzó a publicar una serie de estudios sobre el póquer en su modalidad Texas Hold'em con límite, sentando las nuevas líneas de investigación en la materia.

Su objetivo principal era (y sigue siendo) crear un programa de póquer capaz de jugar mejor que cualquier humano, y de paso aprovechar el póquer como plataforma experimental para realizar avances en la IA.

La primera publicación del grupo (15) sobre este tema fue simplemente una investigación sobre los trabajos previos, con una serie de recomendaciones para el desarrollo de programas de póquer en el ámbito académico.

Más adelante crearon su primer programa de póquer (16), *Loki*, compuesto de tres módulos principales:

- Un evaluador de manos, para determinar el valor de sus cartas y estimar la probabilidad ganar la mano.
- Una estrategia de apuestas basada en sistemas expertos, que elige la acción a tomar a partir del estado del juego y el valor estimado de las cartas que lleva.
- Un modelador de oponentes, que utiliza el historial de juego del oponente para predecir su juego.

El programa no consigue obtener un gran éxito en partidas contra jugadores humanos en lo que se refiere al modelado de oponentes, ya que no está hecho para enfrentarse a jugadores que cambian su estilo de juego durante la partida. Además su estrategia de apuestas basada principalmente en reglas lo convierte en un jugador muy débil frente a cambios en el juego. Sin embargo este primer enfoque, aunque mayormente intuitivo, obtiene resultados interesantes en el rendimiento de su evaluador de manos,

creado a partir de un vector con todas las posibles combinaciones de cartas, a las que se les asigna un peso, que determina la probabilidad de que el oponente las tenga. Entonces dicha probabilidad va cambiando su valor en cada turno de apuestas en función de las cartas comunitarias, el número de veces que se apostó, etc.

Posteriormente en (17) se le agrega un simulador que evalúa las posibles formas en que se puede desarrollar la mano en juego, obteniendo información valiosa para la posterior toma de decisiones y sentando así el modelo (que puede verse en la Figura 1) para futuros desarrollos de agentes.

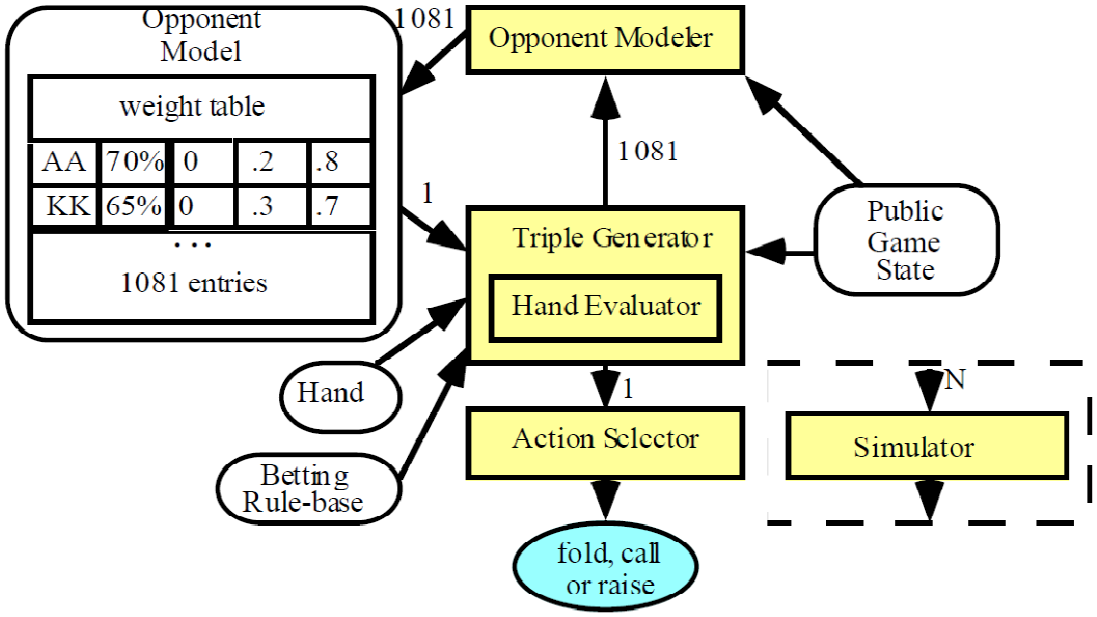



Figura 1: Arquitectura de Loki en (18)

Nuevas características fueron añadidas a *Loki* en (19), entre ellas un nuevo componente para añadir impredecibilidad a la toma de decisiones, los tríos de probabilidades, consistentes en una estructura de datos con 3 números que representan la probabilidad de que se lleven a cabo cada una de las 3 acciones posibles en el póquer (abandonar, pasar o igualar y subir).

Por ejemplo:

*Supongamos que el programa ha hecho todos los cálculos necesarios y genera el trío [0.3, 0.2, 0.5], esto indicaría que hay un 30% de probabilidad de que el programa abandone, un 20% de que pase o iguale la jugada y un 50% de que suba. Finalmente el selector de acciones generaría un número aleatorio entre 0 y 1 que determinaría la opción elegida.*

Ese mismo año el programa se reescribe y es bautizado como *Poki* (20), con una arquitectura prácticamente idéntica pero incorporando un importante avance en el modelado de oponentes, pues ahora la estructura encargada de almacenar el modelo del oponente es una tabla donde se almacena el número de veces que el rival hace cada jugada. Este tipo de modelado fue escogida gracias a un estudio preliminar (21) en que se utilizaban redes de neuronas artificiales (RNA) para predecir las acciones del oponente basándose en un historial de unos cientos de manos.

El estudio reveló que los dos componentes más importantes a la hora de predecir la siguiente acción del rival son las acciones anteriores y la cantidad apostada en ellas. Los resultados fueron bastante buenos ya que la estrategia implementada con la información obtenida de la RNA generalmente conseguía adivinar un 80% de las acciones del rival.

Sin embargo el avance más significativo se produce al introducir un nuevo método de búsqueda en árboles para juegos de información imperfecta: el algoritmo Miximax. Estudiado en profundidad en (22), dicho algoritmo se introdujo con la idea de encontrar un método más preciso de calcular el valor esperado (VE) de cada una de las acciones posibles del programa.

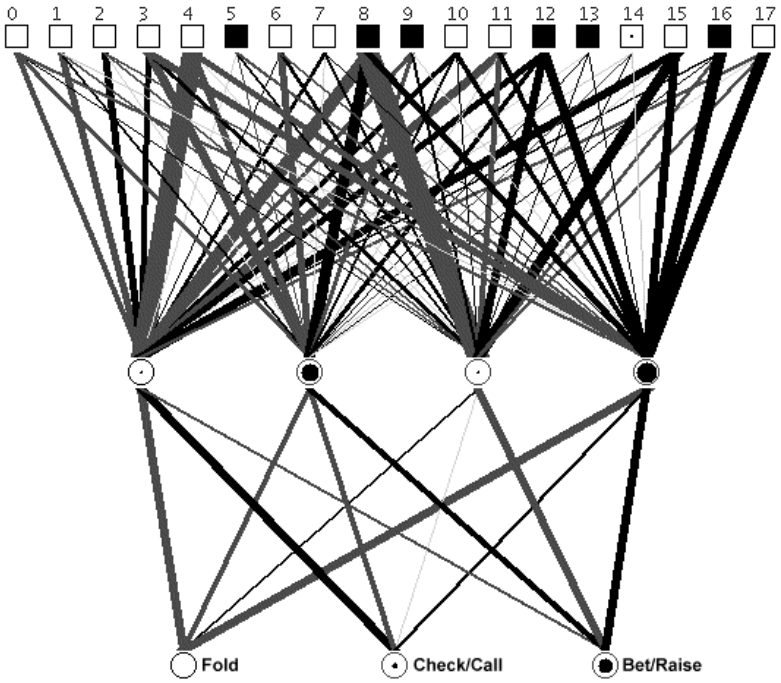



Figura 2: RNA prediciendo la futura acción del rival en (20)

Con esta aportación se consigue hacer una búsqueda completa, hasta los nodos hoja[7], en árboles de juegos con un componente estocástico y donde existe información oculta a los jugadores, es decir, el algoritmo *Miximax* se podría considerar como una adaptación del algoritmo *Expectimax*[8] al póquer.

Otra de las novedades de *Miximax* es que utiliza un histograma de frecuencias para determinar las veces que el oponente ha mostrado los distintos tipos de cartas. Aunque se suele utilizar un histograma de 20 celdas, éste puede ser de otro tamaño, siempre teniendo en cuenta que a mayor número de celdas, mayor precisión, pero también mayor es el número de datos necesarios para que la muestra sea útil.

Por ejemplo, si usamos un histograma de 10 celdas significa que dividiremos los tipos de manos en 10 categorías distintas en función de su valor. Imaginemos entonces que en la mano anterior el oponente mostró A♥ Q♥ y las cartas comunitarias fueron 2♥ 3♥ 7♦ 9♣ J♥, en este caso se le asignaría un valor 10 (Asumimos que el lector conoce las posibles combinaciones de manos del Texas Hold'em junto con su valor, así como la

[7] Aquellos nodos que no tienen hijos y por tanto determinan los estados finales del juego.
[8] *Expectimax* es el algoritmo de búsqueda homólogo a *Minimax* pero para dominios con un elemento estocástico.

terminología común empleada en el juego. En caso contrario se ha incluido un anexo con dicha información) y se incrementaría en 1 el número correspondiente a ése valor, por ejemplo un histograma [0 1 0 5 0 0 0 1 2 2] pasaría a ser [0 1 0 5 0 0 0 1 2 3].

Un ejemplo de cómo se puede aplicar *Miximax* para calcular el VE de un nodo sería el siguiente:

*Supongamos que estamos en la última ronda de apuestas e inicialmente había un bote de 4 ciegas grandes o bb[9], entonces apostamos 2bb y el rival pone las 2bb y nos sube otras 2bb (subiendo el bote total a 10bb).*

*Ahora tenemos que decidir si abandonar (a), igualar (g) o subir (s) de nuevo, calculando cuál de estas opciones tiene mayor VE porque será la que más ganancias nos va a reportar a la larga.*

*Para ello lo ideal sería saber qué mano lleva el oponente, pero como no es posible determinarlo con exactitud, lo que hacemos es hallar su rango de manos, es decir, el tipo de manos que suele llevar en situaciones de juego similares. Esto se consigue mirando el histograma de frecuencias de esa situación de juego. Supongamos que en este caso el histograma es [2 1 0 0 0 0 0 4 3 0] y nuestra mano tiene un valor de 7, esto quiere decir que, basándonos en las veces que el oponente enseñó su mano, hay un 30% de posibilidades de que le ganemos, porque el oponente enseñó 2+1=3 veces cartas a las que vamos ganando, y un 70% de que tenga cartas que nos ganen, porque el oponente enseñó 4+3=7 veces cartas que nos ganarían.*

*Sabiendo que la fórmula para calcular el VE de una acción (A) es:*

$$VE_A = \sum_{i \epsilon \{a,g,s\}} P(i|A) * VE(i|A)$$

*Exceptuando para las acciones que finalizan la mano (nodos hoja, H), en cuyo caso el VE es:*

[9] Del ingles 'big blinds'

$$VE_H = P_{ganar} * B - C$$

*Donde $P_{ganar}$ es la probabilidad de ganar, B es el bote que se ganaría y C el coste de alcanzar dicho nodo hoja.*

*Ahora procedemos a calcular los VE de cada jugada (las letras en minúscula indican acciones del agente y aquellas en mayúscula indican acciones del rival):*

1.1. $VE(a) = \mathbf{0bb}$

1.2. $VE(g) = 0{,}3 * 12 - 2 = +\mathbf{1.6bb}$

1.3. $VE(s)$*: en este caso sabemos que cuando hemos resubido otras 2bb el rival:*

1.3.1. *A abandonado un 20% de las veces y hemos ganado el bote*

1.3.2. *A igualado un 80% y su histograma era [0 0 0 0 0 0 0 5 5 0] con lo que ganaría a nuestra mano actual.*

1.3.3. *Nunca ha resubido de nuevo.*

*Por lo tanto:*

$$VE(s) = 0{,}2 * VE(sA) + 0{,}8 * VE(sG) + 0 * VE(sS)$$

*Donde:*

$$VE(sA) = 1 * 14 - 2$$

$$VE(sG) = 0 * 16 - 2$$

$$VE(sS) = P_{sSa} * VE(sSa) + P_{sSg} * VE(sSg) + P_{sSs} * VE(sSs)$$

*(No sería necesario estudiar este último caso ya que no se ha dado nunca)*

*Luego:*

$$VE(s) = 0{,}2 * [1 * 14 - 2] + 0{,}8 * [0 * 16 - 2] = +\mathbf{0,8bb}$$

*Concluimos que deberíamos sólo igualar la subida del rival ya que es la jugada que a la larga más expectativa positiva nos dará. En* la *Figura 3 se muestra el árbol con el cálculo realizado.*

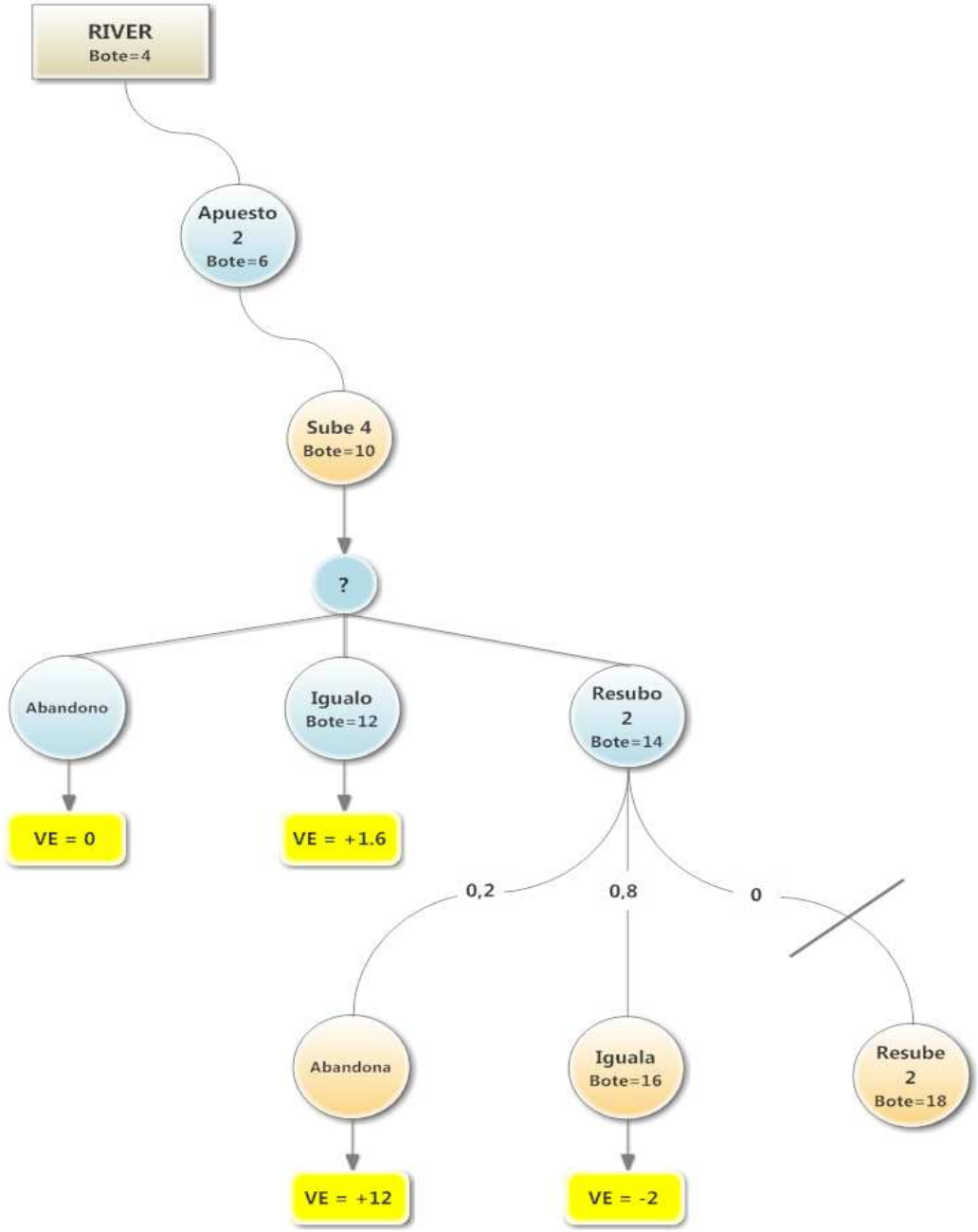



Figura 3: Ejemplo de cálculo del VE de las distintas jugadas posibles

Nuevos avances basados en el algoritmo *Miximax* dieron como resultado la creación de dos nuevos programas, *PsOpti* y *Vexbot*, que al contrario que su predecesor *Poki*, ahora conseguían mejorar significativamente sus resultados en partidas de dos jugadores.

## 2.4. Los Campeonatos de Póquer Hombre-Maquina

En 2007 el CPRG desarrolló *Polaris*, que utiliza un sistema multiagente, con el que compitió ese mismo año en la primera edición del Campeonato de Póquer Hombre-Máquina (Man-Machine Poker Competition).

Se trataba de la primera vez que un agente de póquer era desafiado por dos jugadores profesionales de talla mundial en un experimento científico controlado y con dinero real. La partida de 50000$ se llevó a cabo en Vancouver, Canadá, durante la AAAI[10] (Asociación para el Avance de la Inteligencia Artificial) de ese mismo año.

La competición consistió en cuatro partidas duplicadas de 500 manos cada una. En cada partida duplicada ambos jugadores terminan jugando las mismas series de cartas, es decir, se realiza una primera mitad de partida normal y en la segunda mitad las cartas que recibió *Polaris* le serán repartidas al jugador humano y las que jugó el humano le serán repartidas a *Polaris*. Éste procedimiento se lleva a cabo para minimizar el factor suerte durante la partida, que podría inclinar la balanza para uno u otro lado hasta el punto en que los resultados pudiesen llegar a no ser relevantes.

El ganador se determinó al final de la partida, sumando el total de fichas ganadas o perdidas por cada equipo. En la primera sesión *Polaris* obtuvo una ventaja de +70$ lo que se consideró como un empate, sin embargo la ventaja fue mayor en la segunda sesión, donde ganó a los dos profesionales con una diferencia de +925$.

La tercera sesión fue para los jugadores humanos ya que *Polaris* perdió por -830$, así que todo se decidió en la cuarta sesión, donde los humanos consiguieron ganar por un pequeño margen, lo que les hizo ganar la competición.

[10] Association for the Advancement of Artificial Intelligence.

Sin embargo esta competición fue un éxito debido a que *Polaris* demostró que los agentes de póquer estaban a la altura de algunos de los mejores jugadores del mundo. De hecho el margen por el que ganaron los jugadores humanos fue muy pequeño.

*Polaris* es un sistema compuesto por varios agentes distintos donde cada uno elige la jugada que cree mejor. La toma de decisiones se realiza entonces combinando las decisiones de todos los agentes. Entre los agentes incluidos se encuentra una implementación del algoritmo CFR[11], cuyo objetivo es buscar el punto de equilibrio de Nash, es decir, hallar la estrategia óptima de juego suponiendo que el oponente desarrollará también una contra estrategia óptima.

En 2008, una versión mejorada de *Polaris* compitió en el Segundo Campeonato de Póquer Hombre-Máquina, esta vez contra seis de los más grandes jugadores humanos en el mundo del póquer online. La partida se llevó a cabo en Las Vegas, con un resultado de 3 victorias, 2 derrotas y 1 empate marcando un hito importante en el desarrollo de jugadores de póquer por ordenador.

## 2.5. La Competición Anual de Póquer por Ordenador

Desde el año 2006 se ha venido realizando la Competición Anual de Póquer por Ordenador o ACPC[12], que históricamente se realiza cada verano durante la conferencia anual de la AAAI. El evento atrae todo tipo de competidores con sus desarrollos de agentes, desde equipos de investigación académica a aficionados en la materia venidos desde todos los países. El objetivo de esta competición es promover y evaluar el desarrollo de la IA a través de los retos que proponen las distintas modalidades de póquer.

De este modo la competición no solamente se centra en quién ha ganado, sino en tratar de hallar por qué ha ganado. Por lo tanto se trata de una competición abierta

[11] Counterfactual Regret Minimization es un algoritmo basado en la teoría de juegos que busca el punto de equilibrio de Nash en juegos con un alto componente estocástico y de información imperfecta
[12] Annual Computer Poker Competition.

donde cada uno de los participantes explica las técnicas y estrategias utilizadas por su agente.

Durante los primeros años de la competición sólo se permitían agentes que jugasen en la modalidad Texas Hold'em con límite para dos jugadores, sin embargo en la actualidad también se juega la modalidad sin límite, incluyendo partidas de dos y tres agentes.

En todas las competiciones ha sido muy destacada la actuación de los agentes presentados por el grupo de la CPRG de la Universidad de Alberta, obteniendo sendos primeros puestos en la mayoría con *Hyperborean*, su agente más avanzado.

39 agentes se presentaron a la última edición, celebrada en el 2011, de los que 21 compitieron en la modalidad de 2 jugadores con límite de apuestas, 8 compitieron en la modalidad sin límite de apuestas en partidas de también 2 jugadores, y 10 compitieron en la modalidad de 3 jugadores con límite de apuestas.

Algunas de las técnicas utilizadas por los 8 agentes presentados en la modalidad Texas Hold'em sin límite consistieron en:

- Sistemas Expertos basados en reglas.
- Sistemas multiagente combinando diferentes estrategias.
- Modelado de oponentes basado en estadísticas simples del rival.
- Sistemas Basados en minería de datos a través de grandes historiales de manos.
- Algoritmo de minimización CFR basado en la búsqueda del equilibro de Nash junto con distintas técnicas para reducir el número de estados a contemplar en el juego.

Esta última técnica es la que ha obtenido mejores resultados. Se puede obtener más información acerca de cómo se ha llevado a cabo en (23), donde se busca adaptar los avances realizados en los agentes que juegan en la modalidad Texas Hold'em con límite a la modalidad de apuestas sin límite, cuya complejidad es mayor debido a que

no solo hay que determinar qué jugada hacer sino que también se debe elegir la cantidad correcta a apostar, por lo tanto el número de estados posibles del juego crece enormemente.

En dicho trabajo se aborda este problema aplicando ciertas simplificaciones al juego para reducir el número de estados posibles y abreviar los cálculos necesarios para evaluar cada jugada. Una de las técnicas se basa en agrupar las cartas en conjuntos que presentan valor y características parecidas. Por ejemplo las manos AA y KK (las de mayor valor preflop en Texas Hold'em) se suelen agrupar en la misma categoría de manos porque la fuerza de ambas es parecida y también la forma en que se juegan.

# Capítulo 3

# El desarrollo de Uc3mBot

En este capítulo se detalla el trabajo llevado a cabo, describiendo el planteamiento, la arquitectura del programa, el diseño de clases y cada uno de los componentes involucrados en el diseño. Para ampliar la información han sido incluidos en la sección de anexos un manual de usuario que explica cómo funciona la aplicación y describe los pasos a seguir para echar una partida contra el programa; y un manual de referencia, de ayuda para aquellos desarrolladores que deseen ampliar el proyecto.

## 3.1. Planteamiento

Bautizado con el nombre de Uc3mbot (bot[13] de la UC3M), el proyecto se ha abordado en dos fases con un desarrollo incremental. El objetivo perseguido con este planteamiento ha sido, por un lado tratar de dividir el problema en partes más simples, de manera que su resolución fuese más abordable, y por otro que se pudiese realizar una primera prueba del programa para así retroalimentar el proceso en el siguiente ciclo.

La primera fase del proyecto se ha centrado en crear una versión inicial del agente de póquer que implementase un sistema experto (reglas creadas a partir de conocimiento

[13] Programa informático que realiza funciones muy diversas, imitando el comportamiento de un humano.

experto) para la toma de decisiones, junto con todos los componentes necesarios para su funcionamiento, como un evaluador de manos y una primera versión de la estructura de datos encargada de almacenar el comportamiento de juego del oponente.

En la segunda fase del desarrollo se han incluido los componentes encargados de modelar el comportamiento del contrincante y de elaborar, a partir de estos datos, las contra estrategias necesarias para maximizar las ganancias, obteniendo el beneficio máximo de los posibles errores en el juego del rival. Para esto inicialmente se pensó en utilizar un programa que recogiese los datos de cada mano jugada y diese unas estadísticas que dibujasen el estilo de juego del oponente. La idea era entonces añadir un conjunto de reglas nuevas al sistema experto ya creado, para que el programa jugase de un modo u otro en función de las estadísticas obtenidas. El problema que tenía este planteamiento es que resultaba muy difícil determinar el tipo de jugada a efectuar basándose sólo en las estadísticas de juego del oponente, ya que para ello se requiere una enorme cantidad de reglas, que a su vez limitan la flexibilidad de juego del programa y en muchos casos lo hacen más predecible.

Se optó por lo tanto por descartar esta opción y tratar de ingeniar en su lugar una estrategia de juego donde el agente tuviese la funcionalidad de crear un modelo del adversario en tiempo real y aprendiese a su vez de dicho modelo.

Con este fin en mente se optó por implementar una versión del algoritmo de búsqueda *Expectimax* adaptado a este juego y estudiar la jugada de mayor expectativa en cada situación. Este cambio requirió transformar la estructura de datos utilizada en la primera versión de Uc3mBot en los actuales *árboles de frecuencias*, que describiremos en detalle más adelante.

Se estableció también necesario añadir a su vez un componente de simulación, capaz de calcular la probabilidad de ganar a largo plazo enfrentando las cartas de Uc3mBot frente al rango de cartas que pudiese llevar el oponente en cada caso. En los siguientes apartados se explica el funcionamiento de cada componente del sistema.

## 3.2. Arquitectura del programa

Uc3mBot es un agente de póquer que funciona como un cliente dentro del entorno de ejecución, ya que se conecta a un servidor de juego, que es el encargado de repartir las cartas y enviar toda la información necesaria acerca del desarrollo de la partida. Para este desarrollo se ha escogido trabajar con el servidor que proporciona la ACPC, ya que implementa un protocolo de comunicación sencillo, es gratuito y es el servidor utilizado para arbitrar las partidas de dicha competición, lo que simplificará una posible futura presentación del agente a dicha competición. Más información sobre el servidor y protocolo de comunicación se puede encontrar en la sección 3.4.1 donde se explica en mayor detalle.

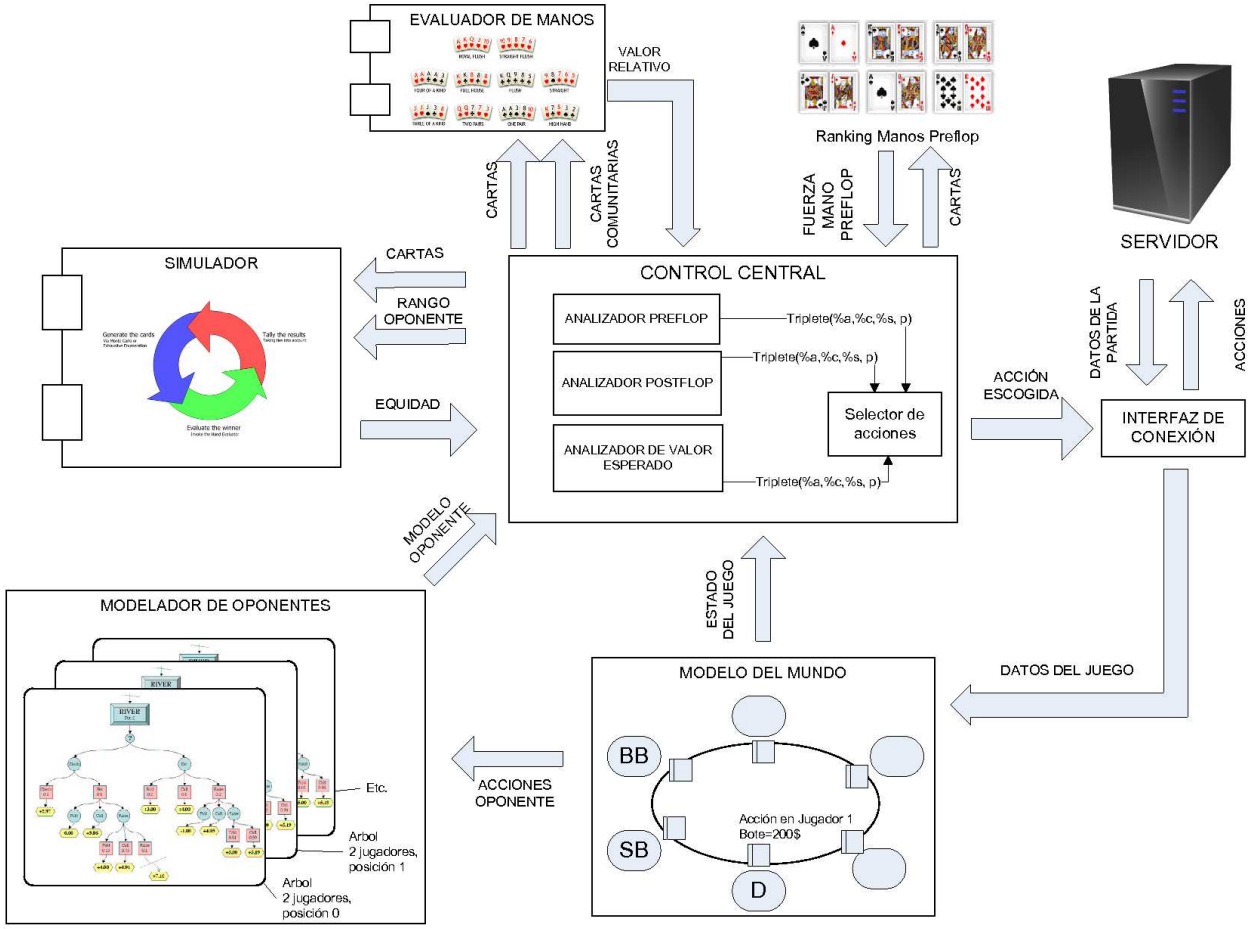



Figura 4: Arquitectura de Uc3mBot

Dentro de Uc3mBot distinguimos varios módulos que trabajan de manera coordinada para tomar la mejor jugada posible en cada situación de juego que el servidor le presenta. En la Figura 4 se muestran dichos módulos y de qué modo intercambian datos durante el análisis de una mano de póquer. A continuación pasamos a explicar cada uno de ellos en detalle.

### 3.2.1. La Interfaz de Conexión

A la hora de crear una partida nueva Uc3mBot se ha de conectar al servidor de juego a través de una nueva conexión TCP/IP al puerto y dirección donde el servidor está escuchando. El proceso de ejecución de cada mano de póquer comienza entonces cuando el servidor envía una cadena con información sobre la una nueva mano. Estos datos contienen además de las cartas repartidas a Uc3mBot, la ronda de juego y las apuestas que se efectuaron. Dado que están en un formato difícil de interpretar y utilizar, los mensajes son traducidos por la interfaz de conexión, que es la encargada a tal efecto.

Este componente no sólo traduce los mensajes que llegan del servidor a Uc3mBot, sino que también es el encargado de traducir las respuestas del agente de nuevo al servidor.

La interfaz de conexión contiene una serie de rutinas ideadas para extraer cada una de las partes en que se divide la información de cada uno de los mensajes recibidos. Este componente es el encargado de identificar toda la información relativa a:

- La ronda en que se encuentra el juego
- El tamaño actual del bote
- La posición de Uc3mBot en la mano
- El número de jugadores en la mano
- Las cartas privadas de Uc3mBot
- Las cartas comunitarias repartidas sobre la mesa
- La secuencia de apuestas efectuada durante la mano
- A quién le toca jugar
- Etc.

Una vez se descifra un mensaje, la información es separada y enviada al componente encargado de actualizar y dar forma al modelo del mundo de Uc3mBot, creando las estructuras de datos necesarias para que el agente pueda jugar dicha mano.

### 3.2.2. El Modelo del Mundo

El modelo del mundo que Uc3mBot almacena consiste de una serie de representaciones internas que le mantienen informado del estado del juego y que le permiten interactuar con éste. Además de mantenerle informado del resultado de todas las manos que se van jugando durante la partida, este componente es el encargado de recoger y dar forma a la información que le envía la interfaz de conexión sobre la posición de cada jugador, las cartas del agente, etc.

También se encarga de actualizar la pila de fichas de los jugadores, calcular el tamaño de la subida (si es que se ha subido) y poner al día la lista de acciones efectuadas en la mano.

Por un lado las jugadas que ha llevado a cabo el oponente se enviarán al *modelador de oponentes* para que éste actualice el *árbol de frecuencias* correspondiente (En la sección3.3 se describe qué es y para qué sirve este árbol), y por otro se envía el resto de información al control central de Uc3mBot, dedicado a la toma de decisiones.

Con el fin de mantener actualizada toda la información durante el juego, éste componente crea una representación interna de la mesa de póquer, incluyendo información sobre las cartas comunitarias, los jugadores y la ronda actual, tal y como se muestra en la Figura 5, donde se ilustra gráficamente el modelo que se crea durante una posible situación de juego.

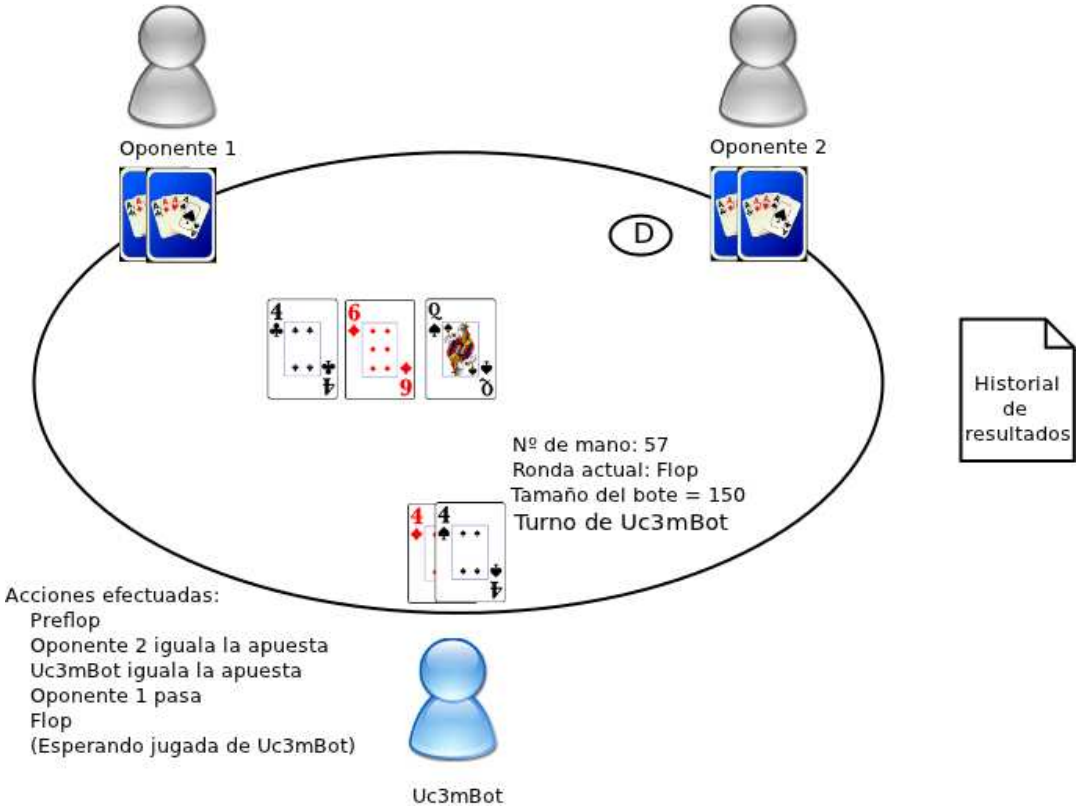



Figura 5: Representación gráfica del modelo del mundo de Uc3mBot

### 3.2.3. El Módulo de Control Central

El módulo principal de Uc3mBot es denominado como control central, pues es el encargado de coordinar las operaciones principales. Una vez este módulo recibe los datos, se encarga de escoger qué analizador es más adecuado para utilizar, pues existen 3 analizadores, dos de ellos creados mediante sistemas expertos apoyados en operaciones pre calculadas, un evaluador de manos y un simulador; y un tercero, el *analizador de valor esperado* o *VE*, dedicado a explotar las debilidades que el rival pudiera tener en su juego.

Al principio de una partida Uc3mBot no tiene información sobre el juego del adversario y por lo tanto no puede determinar aún qué jugada tendrá mayor expectativa positiva, por éste motivo y por el hecho de que es importante mezclar el juego desarrollado para no resultar demasiado predecible, han sido creados el *Analizador Preflop* y el

*Analizador Postflop*, el primero dedicado exclusivamente a analizar situaciones en la ronda preflop[14] (antes de las cartas comunitarias) y otro dedicado a las demás rondas.

Por ejemplo el *analizador preflop* utiliza un ranking que contiene todas las posibles combinaciones de cartas que se pueden repartir a un jugador, ordenadas de mayor a menor valor esperado. Dicho ranking fue generado mediante un algoritmo que escogía cada uno de los posibles pares de cartas y, a través de multitud de simulaciones, las enfrentaba a todas las demás combinaciones en partidas de dos jugadores, obteniendo así un valor esperado (VE) medio para cada una de ellas. Después el ranking se ordenó con un algoritmo Quicksort.

Con este ranking Uc3mBot puede medir en la ronda preflop la fuerza de las cartas que le han repartido y así actuar en consecuencia. Por regla general cuanto mayor sea el valor esperado de unas cartas, más subidas tenderá a realizar. En la Figura 6 se muestra una pequeña parte del ranking generado.

Éste analizador junto con el Analizador Postflop se pondrá en funcionamiento mientras no se tengan suficientes muestras del oponente, como para empezar a realizar cálculos sobre el VE de cada jugada con suficiente precisión, ya que el último analizador, el analizador de VE, requiere de un mínimo de muestras para funcionar correctamente.

[14] En el Texas Hold'em existen cuatro rondas de juego: preflop, flop, turn y river, donde la primera es la única que se juega sin cartas comunitarias. Ver anexo con las reglas del juego.

```
final static Cartas lista[] = {

                    new Cartas("AhAc", 0.8520371330210104),
                    new Cartas("AdAc", 0.8520371330210104),
                    new Cartas("AdAh", 0.8520371330210104),
                    new Cartas("AcAs", 0.8520371330210104),
                    new Cartas("AhAs", 0.8520371330210104),
                    new Cartas("AdAs", 0.8520371330210104),
                    new Cartas("KcKs", 0.823956797867859),
                    new Cartas("KhKs", 0.823956797867859),
                    new Cartas("KdKc", 0.823956797867859),
                    new Cartas("KdKh", 0.823956797867859),
                    new Cartas("KdKs", 0.823956797867859),
                    new Cartas("KhKc", 0.823956797867859),
                    new Cartas("QhQc", 0.7992516406108319),
                    new Cartas("QdQc", 0.7992516406108319),
                    new Cartas("QdQh", 0.7992516406108319),
                    new Cartas("QcQs", 0.7992516406108319),
                    new Cartas("QhQs", 0.7992516406108319),
                    new Cartas("QdQs", 0.7992516406108319),
                    new Cartas("JhJc", 0.7746947290114992),
                    new Cartas("JdJc", 0.7746947290114992),
                    new Cartas("JdJh", 0.7746947290114992),
                    new Cartas("JcJs", 0.7746947290114992),
                    new Cartas("JhJs", 0.7746947290114992),
                    new Cartas("JdJs", 0.7746947290114992),
                    new Cartas("TdTs", 0.7501177995095664),
                    new Cartas("ThTc", 0.7501177995095664),
                    new Cartas("TcTs", 0.7501177995095664),
                    new Cartas("ThTs", 0.7501177995095664),
                    new Cartas("TdTh", 0.7501177995095664),
                    new Cartas("TdTc", 0.7501177995095664),
                    new Cartas("9h9c", 0.7205725194515336),
                    new Cartas("9d9h", 0.7205725194515336),
                    new Cartas("9d9c", 0.7205725194515336),
                    new Cartas("9h9s", 0.7205725194515336),
                    new Cartas("9c9s", 0.7205725194515336),
                    new Cartas("9d9s", 0.7205725194515336),
                    new Cartas("8h8c", 0.6916303546900217),
                    new Cartas("8d8h", 0.6916303546900217),
                    new Cartas("8d8c", 0.6916303546900217),
```

Figura 6: Muestra del ranking de cartas preflop ordenado por valor esperado

Para determinar si se puede utilizar el analizador VE o no, el control central cuenta el número de veces que el rival enseñó sus cartas en una mano anterior con una situación de juego semejante (con la misma estructura de apuestas), es decir, se determina un umbral N (cuyo valor se puede ajustar), y si el oponente enseñó sus cartas un número de veces mayor o igual que N se utilizará el dicho analizador, que escoge una jugada u otra buscando obtener el máximo de ganancias a partir del estilo de juego del rival. Mientras ése número de muestras no se alcance, se aplicarán los otros dos analizadores.

Las decisiones que toman cada uno de los analizadores se plasman en un trío de probabilidades, que es una estructura de datos que sirve para introducir

impredecibilidad en la toma de decisiones. Dicha estructura es semejante a la utilizada en el desarrollo de otros agentes, salvo que en este caso se le ha añadido un componente más, denominado *factor de subida* que determina el tamaño de la apuesta, ya que no es un tamaño fijo como en la modalidad con límite.

En consecuencia el trío de probabilidades tiene la siguiente forma:

$$[P_a, P_c, P_s, F]$$

Donde,
Pa: Probabilidad de abandonar
Pc: Probabilidad de pasar o igualar
Ps: Probabilidad de subir
F: Factor de subida

Por ejemplo si uno de los analizadores diese el trío [0.2, 0.4, 0.4, 2] significaría que el 20% de las veces debería abandonar, el 40% de las veces debería pasar o igualar la apuesta y el otro 40% restante debería subir una cantidad igual a 2 veces el tamaño actual del bote.

No hay que olvidar que para que este modo de toma de decisiones tenga sentido es necesario que se cumpla que:

$$P_a + P_c + P_s = 1$$

Es importante señalar que la mayor diferencia entre el póquer con límite y el póquer sin límite está en que en éste último es necesario determinar el tamaño de apuesta que se quiere realizar no siendo así en las modalidades con límite de apuestas. Ésta característica añade una serie de dificultades importantes, porque según el tamaño de la apuesta realizada se pueden clasificar muchas más jugadas distintas que las que se emplean en el póquer con límite, donde sólo se puede abandonar, pasar/igualar o subir. Sin embargo en el póquer sin límite se puede subir todo tipo de cantidades, desde el doble de la última apuesta hasta envidarse (apostar todo) además de abandonar y pasar/igualar.

Finalmente el proceso de la toma de decisiones termina cuando el control central envía la acción tomada al servidor a través de la interfaz de conexión, que es quien traduce las decisiones junto con los datos necesarios al lenguaje del servidor.

### 3.2.4. El Modelador de Oponentes

Era primordial para el funcionamiento de este agente utilizar una estructura de datos que de alguna forma, almacenase toda la información necesaria acerca del juego de cada uno de los oponentes en la mesa, con la idea de ser explotada por los analizadores incluidos en Uc3mbot. Distinguiendo además entre la forma de jugar desde las distintas posiciones que ofrece la mesa. Ahí es donde entra en juego el modelador de oponentes.

Este componente está formado por una representación interna de cada oponente, donde se almacenan sus jugadas en diferentes estructuras en forma de árbol, a las que llamaremos *árboles de frecuencias*.

Cada oponente tendrá en su modelo distintos árboles de frecuencias, guardando cada uno las jugadas realizadas desde una determinada posición en la mano. Por ejemplo, en una partida de sólo dos jugadores la clase Oponente almacenará dos árboles, uno de ellos representará las manos en que el oponente jugó desde la posición 0 (ciega grande) y el otro las manos que jugó desde la posición 1 (ciega pequeña o repartidor). En una partida de tres jugadores Uc3mBot creará tres árboles para cada oponente, uno de ellos representará las manos en que el oponente jugó desde la posición 0 (ciega pequeña), el otro las manos que jugó desde la posición 1 (ciega grande), y el último las manos que jugó desde la posición 2 (repartidor).

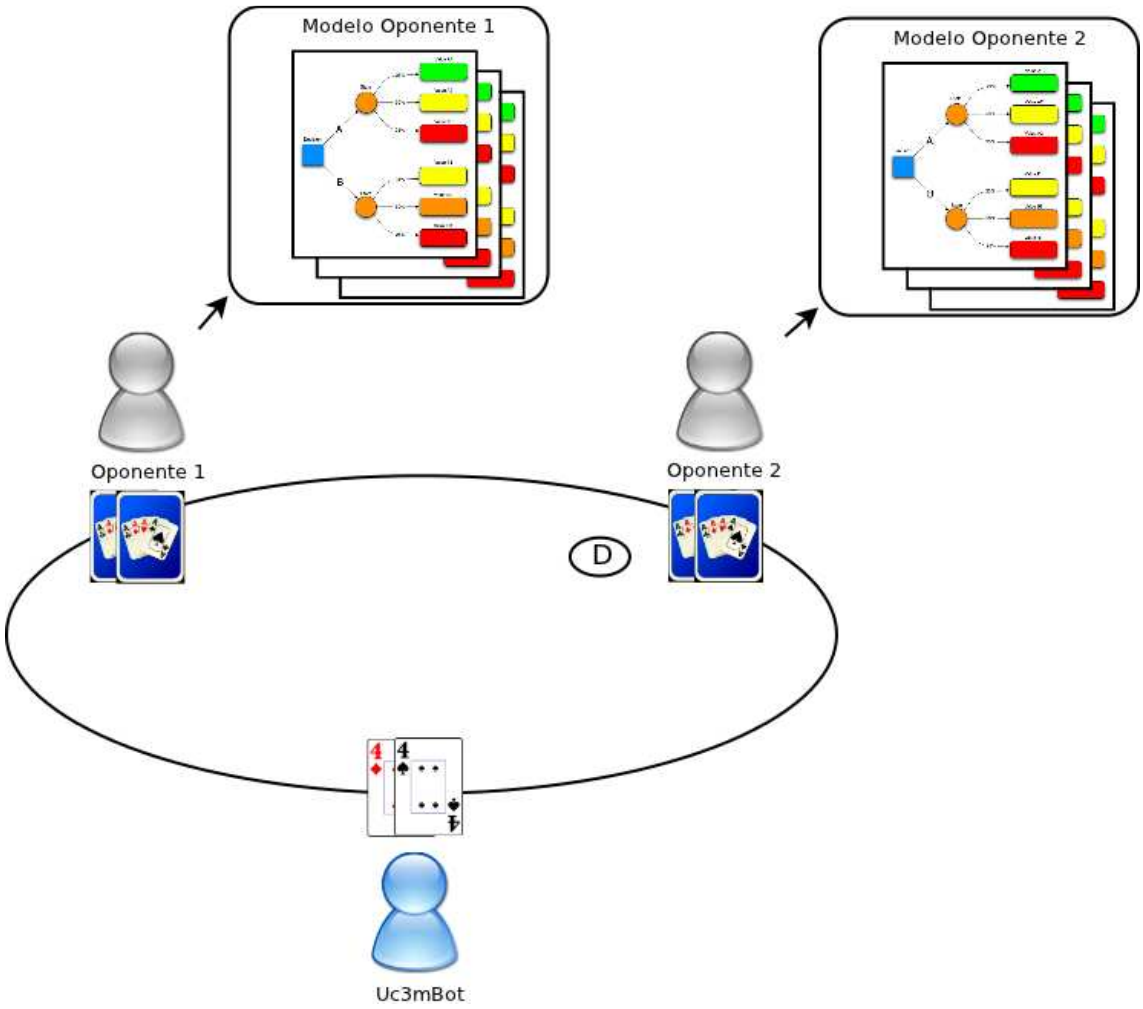



Figura 7: Representación del modelo de cada oponente en una mesa de tres jugadores

De este modo se crearán tantos árboles como posiciones diferentes haya en la mesa, esto se ha hecho así debido a que la forma de jugar cambia considerablemente dependiendo del número de jugadores y de la posición que se ocupa en la mano.

### 3.2.5. El Evaluador de Manos

El analizador de las rondas flop, turn y river, al que hemos nombrado *analizador postflop*, necesita calcular el valor de la mano en tiempo real, ya que no todos los tipos de jugada son posibles pues todo dependerá de qué cartas comunitarias se reparten sobre la mesa en cada caso, y además el rango de manos de cada oponente irá cambiando según se desarrolla la partida.

Por ejemplo no es posible ligar un póquer con un flop de tipo 4♥ 2♦ Q♣ donde no hay carta doblada, por tanto ésa jugada habría que descartarla en este caso y tener en cuenta que la jugada máxima con este flop es un trío de damas.

Para calcular el valor de la mano, el analizador postflop realiza por lo tanto llamadas al evaluador de manos y al simulador. Estos dos componentes le proporcionarán entonces los datos necesarios para estimar un número de apuestas concreto, que Uc3mBot deberá tratar de conseguir durante el desarrollo de la mano. Cuanto mayor sea el valor de nuestras cartas en relación a todas las demás combinaciones, mayor valor intentaremos sacarle a la mano.

La función del evaluador de manos es entonces calcular el valor absoluto de las cartas de Uc3mBot y de todas las demás combinaciones posibles. A continuación con este dato, el analizador determina el valor relativo a todas las demás combinaciones (mediante nuevas llamadas al evaluador), con esto se consigue determinar con precisión el número de combinaciones de cartas del total que ganan a su mano.

El evaluador de manos funciona mediante una librería externa llamada Poker-eval que pertenece al proyecto de código abierto Pokersource. Poker-eval es una librería en C que evalúa manos de póquer. El resultado de cada evaluación es un número que determina el valor de la mejor combinación que se puede hacer con las cartas de Uc3mBot. La idea general es que si la evaluación de tu mano es menor que la evaluación de la mano de tu oponente, pierdes. Sin embargo, con el propósito de facilitar la interpretación de los datos, se ha añadido un método en este proyecto que traduce el valor numérico de las cartas a un formato inteligible, de tal forma que por ejemplo un valor de 33620737 se traduciría como unas dobles parejas.

La principal ventaja de esta librería es que está diseñada para ser rápida, lo cual permite realizar un mayor número de cálculos durante el turno. Poker-eval se basa en dos estrategias para maximizar la velocidad de procesamiento:

1. Manipulación a nivel de bits para la representación de las cartas.
2. Tablas de consulta con los valores precalculados.

Una mano de póquer es representada internamente por Poker-eval como una secuencia de 52 bits, utilizando un entero de 64 bits para almacenarla. El formato es el siguiente:


Usando este tipo de representación una mano como la escalera real de corazones sería:

**00000000 00000000 00000000 00000000 00000000 00000000 11111000 00000000**

En cuanto a las tablas de consulta, son estructuras de datos en forma de array que se usan para sustituir rutinas de computación y así ahorrar tiempo de procesamiento durante la partida. Consisten en calcular por adelantado todos los posibles resultados de una función y almacenarlos en una tabla, de manera que para hallar un resultado simplemente basta con buscarlo en la tabla, en vez de realizar todo el cálculo necesario.

### 3.2.6. El Simulador

La función del simulador es estimar con la mayor precisión posible las probabilidades de ganar la mano. Con este fin realiza miles de desarrollos de juego y situaciones posibles a partir del estado de juego actual, enfrentando nuestras cartas a las posibles cartas que pueda llevar el rival, lo que conlleva numerosos cálculos, que en el caso de que el rango de cartas del oponente sea muy amplio, pueden resultar en un incremento importante del tiempo de juego.

Más concretamente el simulador es una calculadora de equidades, que permite enfrentar las cartas de Uc3mBot contra un rango de cartas estimado y que puede funcionar mediante dos métodos distintos:

a) Enumeración exhaustiva
b) Monte Carlo

La enumeración exhaustiva es un caso de combinatoria aplicada que explora todas y cada una de las posibles opciones en que se puede desarrollar la mano con un error de cero, pero como contra partida en la mayoría de ocasiones requiere de una gran cantidad de tiempo para realizar los cálculos.

Monte Carlo es un método estadístico que permite aproximar con un pequeño margen de error los valores obtenidos por enumeración exhaustiva. Consiste en explorar desarrollos de la mano de forma aleatoria, y es muy útil cuando el número de combinaciones a explorar es enorme, pues reduce en gran medida el tiempo requerido para los cálculos sin perder demasiada precisión, ya que los valores obtenidos se aproximan mucho a los valores reales, hasta el punto en que en la práctica dicha diferencia puede ser ignorada.

En caso de que la potencia de computación fuese infinita siempre utilizaríamos el método de enumeración exhaustiva, sin embargo en el mundo real dicha técnica puede resultar prohibitiva en términos de tiempo a medida que crece el número de casos a explorar, de manera que se ha optado en este proyecto por utilizar el método de Monte Carlo en la mayoría de casos.


Figura 8: Funcionamiento del simulador

El simulador funciona mediante un bucle de 3 pasos que se repiten constantemente, llegando a veces a necesitar evaluar millones de partidas repitiendo los pasos de la Figura 8.

## 3.3. El Modelo de Conocimiento

Esta sección expone el modelo de conocimiento en un diagrama de clases, utilizando la notación UML 2.0, que es la más utilizada a la hora de describir un programa creado con un lenguaje de programación orientado a objetos (POO), como es Java. El modelo fue creado durante las fases de análisis y diseño de Uc3mBot, donde se establecieron los componentes que se encargarían del funcionamiento y la información que se manejaría en el sistema. Cabe señalar que se ha ido ampliando en función de las modificaciones y mejoras adoptadas durante la fase de implementación.

A continuación se describen las clases más importantes para el funcionamiento de Uc3mBot junto con sus relaciones respectivas, para más información se puede consultar el diagrama de clases completo en el 0 al final del documento.

La clase **Cliente** es la parte de la aplicación que se ejecuta primero ya que es la interfaz de conexión de Uc3mBot con el servidor. Tiene dos clases hijas que son ClienteACPCv1 y ClienteACPCv2, creadas para conectarse respectivamente con las versiones 1 y 2 de los servidores aportados por la ACPC. Los métodos más importantes de ambas clases son aquellos encargados de enviar/recibir mensajes y de transformarlos para que Uc3mBot pueda interpretarlos creando un modelo interno del juego. Los constructores de las clases Cliente necesitan recibir la dirección IP y el puerto de la máquina en que se está ejecutando el servidor, para comenzar a enviar y recibir mensajes de éste. En caso de conectarse a un servidor de la ACPC el primer mensaje determina qué versión del protocolo se va a utilizar.

Una vez conectado al servidor, el cliente llama a gestionarCambioDeEstado() cada vez que recibe un mensaje nuevo de éste. Dicho método se puede considerar como el más importante de esta clase, porque es el encargado de comunicar a la clase Mesa todos los cambios que vayan sucediéndose durante la partida para que ésta vaya creando el

modelo interno del juego, necesario para que nuestro agente pueda acceder a la situación actual de juego en cada instante. Lo primero que este método realiza es ver si ha comenzado una nueva mano, después extrae las cartas comunitarias del mensaje junto con las cartas privadas y lo envía también.

gestionarCambioDeEstado() es el método que determina también de quién es el turno y lo indica, así como la ronda actual o cuál fue la última jugada. También estudia cada mensaje recibido del servidor para saber si la mano ha finalizado o no, e informar así de ello a la clase Mesa.

Para que las clases Mesa y Cliente puedan intercambiar mensajes, ambas contienen una referencia a la instancia de la otra, es decir, la clase Mesa contiene el objeto Cliente que a su vez contiene el objeto de tipo Mesa, creando una relación uno a uno como se aprecia en la Figura 9.

La clase **Mesa** se crea a partir del número de jugadores que van a jugar en ella, las posiciones iniciales del repartidor y de Uc3mBot en la mesa, el tamaño de las ciegas, la pila de fichas inicial y una referencia al objeto Cliente. Esta clase se encarga de mantener actualizadas las estructuras de datos que representan el estado del juego. Su método principal es nuevaJugada(), cuya misión es añadir la última jugada a la lista de acciones efectuadas durante la mano actual, registrar la acción en el árbol de frecuencias correspondiente y actualizar otra información importante como las pilas de fichas de los jugadores. Al igual que una mesa real de juego, esta clase mantiene información de las cartas comunitarias, de la situación de la mano y de los jugadores.

Al finalizar una mano, la clase Mesa se encarga de informar del resultado a Uc3mBot y también de registrarlo en el nodo del árbol de frecuencias correspondiente, además de almacenar las ganancias o pérdidas de ésta en el fichero “historial.txt” junto con la fecha en que se inició la partida.

Toda la información del estado de juego es proporcionada por esta clase, desde la posición que ocupa cada jugador y sus acciones hasta las cartas comunitarias, por lo tanto esta clase es clave para los analizadores a lo hora de estudiar cada situación de la partida.

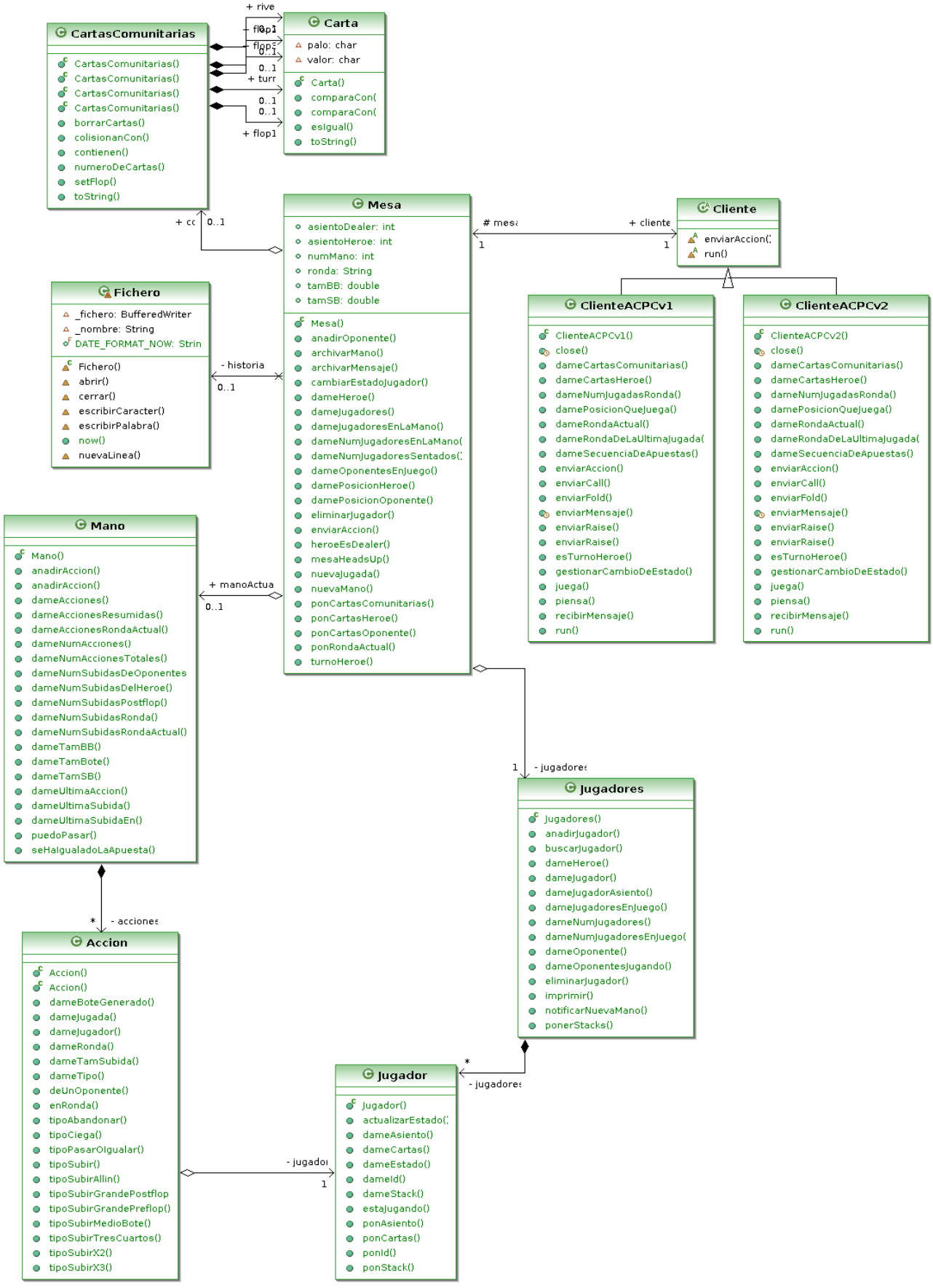



Figura 9: Diagrama de las clases que conforman el modelo del mundo y la interfaz de conexión

Pasamos ahora a hablar de otra de las clases más importantes, la clase **Uc3mBot**, que es la que representa al agente dentro del juego. La clase Uc3mBot es representada como un tipo de jugador para poder ubicarlo dentro del modelo interno del mundo que tiene el agente, como se muestra en la Figura 10.

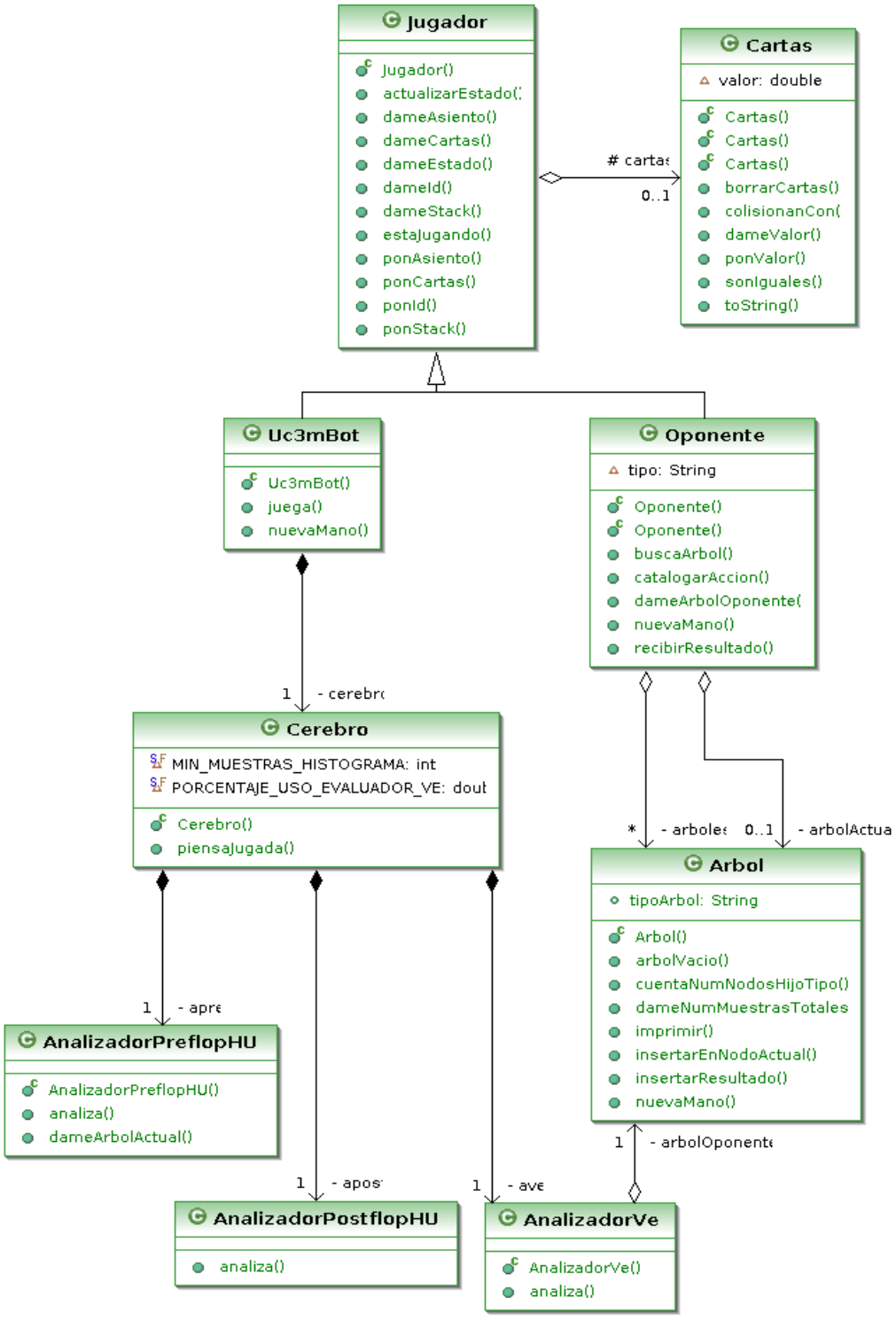



Figura 10: Diagrama de las clases que representan a los jugadores

Uc3mBot contiene los métodos juega() y nuevaMano(). A través del primer método se le informa de que es su turno y debe elegir una acción. Con el segundo método se le informa de que ha comenzado una nueva mano.
Uc3mBot tiene un atributo principal de la clase Cerebro, encargado de escoger el analizador de juego necesario en cada situación. Este atributo utiliza tres analizadores diferentes que selecciona en función de la cantidad de datos que tiene acerca del rival, para ello contiene una serie de constantes que determinan los umbrales a partir de los que escoge uno u otro. Además se encarga de mezclar las distintas estrategias de juego con un porcentaje de aleatoriedad determinado con el objetivo de no desplegar un juego demasiado predecible. Todo esto lo realiza el método piensaJugada() que devuelve un objeto de la clase Triplete que contiene el trío de probabilidades que determinarán la jugada a efectuar.

En realidad todos los cálculos importantes se realizan por los tres analizadores representados por las clases AnalizadorPreflopHU, AnalizadorPostflopHU y AnalizadorVe. Los dos primeros utilizan un sistema experto para la toma de decisiones junto con una librería de evaluación de manos, mientras que el tercero emplea un algoritmo de búsqueda inteligente que estudia las distintas jugadas del oponente para hallar la acción de mayor expectativa matemática. Todos los analizadores requieren de la situación de juego actual, que reciben a través de un objeto de la clase Mesa.

**AnalizadorPreflopHU** tiene como método principal pensarJugadaConEquidad(), que tal como indica su nombre, determina la jugada a realizar hallando la equidad de su mano contra todas las demás manos que pueda llevar el oponente. Como éste cálculo requiere mucho tiempo de procesamiento para ser llevado a cabo debido al gran número de simulaciones necesarias, se ha creado una lista que contiene todas las posibles manos junto con su equidad. Dicha lista se generó enfrentando cada una de las manos contra todas las demás combinaciones posibles utilizando el simulador y un algoritmo Quicksort para ordenarlas después de mayor equidad a menor. De este modo la lista se ha incluido directamente en el código tal y como se muestra en la **¡Error! No se encuentra el origen de la referencia.**.

Una vez determinada la equidad de su mano calcula qué porcentaje de veces es

Una vez determinada la equidad de su mano calcula qué porcentaje de veces es correcto subir, y en caso de haberse generado un gran bote estima lo rentable que resulta seguir jugando. También establece un umbral de equidad sobre el que hacer all-in o no. Dicho umbral va modificándose a medida que se desarrolla el juego, por ejemplo en el caso de que el oponente sea muy agresivo y suba muchas veces preflop el rango de manos con el que Uc3mBot responderá a dicha agresión será cada vez más amplio, siempre intentando tener mejores cartas que el rival a la hora de envidarse.

**AnalizadorPostflopHU** funciona de forma distinta, calcula el porcentaje de cartas que le ganan en una situación concreta (postflop) para así estimar la fuerza de su mano, y con esta información determina un número de apuestas a efectuar que distribuye durante la ronda actual. En el caso de que el número total de apuestas supere al estimado como correcto el analizador se planteará abandonar la mano ya que se creerá perdedor.

Cuantas menos fichas le queden a Uc3mBot en relación con el tamaño del bote, más difícil será que abandone la mano, pues en la mayoría de estos casos es matemáticamente incorrecto abandonar.

El AnalizadorPostflopHU está diseñado para apostar y subir de farol un porcentaje de veces determinado con la idea de introducir un componente de engaño, necesario en toda estrategia que pretenda rivalizar contra jugadores avanzados. Este analizador también realiza apuestas de continuación, que consisten en apostar de nuevo en el flop cuando se ha realizado la última apuesta de la ronda preflop, ya sea habiendo ligado jugada o no.

El cálculo de la fuerza de la mano es llevado a cabo por la clase **EvaluadorDeManos** a través de la librería Poker-eval de código abierto y sobre la que se habló más detalladamente en la sección 3.2.5. El método más importante de esta clase es calcularValor(), ya que es el método sobre el que se apoyan los demás para realizar las operaciones.

El último analizador, el **AnalizadorVE** (analizador de valor esperado), se puede utilizar en todas las rondas de una mano, con la condición de se trate de situaciones de juego

donde el rival terminó por enseñar sus cartas, porque el cálculo del VE para una situación de juego concreta requiere de la estimación del rango de cartas del oponente para un funcionamiento preciso. Éste analizador calcula las probabilidades de tomar una jugada u otra a partir del valor esperado (promedio de ganancias o de pérdidas que podemos obtener si la ejecutamos) de cada una de las posibles jugadas.

Por ejemplo, si no sabemos si decidirnos por igualar una apuesta o abandonar, en el momento en que calculemos el valor esperado de ambas opciones sabremos con certeza que aquella cuyo VE sea mayor es la acción que a largo plazo reportará más beneficios. Un ejemplo de cálculo del VE se muestra en la Figura 3 (apartado 2.3).

Otro ejemplo se puede extraer del juego de la ruleta americana:

> *Este juego que consta de 38 casillas equiprobables donde si apostamos 1 ficha a un solo número y acertamos, nos pagarán 36 fichas. Por lo tanto las ganancias serán de 35 veces lo que hemos apostado (cobramos 36, pero las ganancias netas son 35 ya que hay que descontar lo que hemos apostado), y cuando perdamos, las pérdidas serán de 1. La probabilidad de ganar es por lo tanto de 1/38 y la probabilidad de perder será de 37/38. Entonces el valor esperado de apostar a un solo número es la probabilidad de ganar multiplicado por las ganancias netas menos la probabilidad de perder multiplicada por las pérdidas:*

$$VE(apostar\ a\ un\ n^{\underline{o}}) = \frac{1}{38} * 35 - \frac{37}{38} * 1 = -0{,}0526$$

> *De manera que cada vez que apostemos a la ruleta americana perderemos de media unos 5 céntimos por cada euro apostado.*

En la fórmula de arriba diferenciamos explícitamente entre ganancias y pérdidas, sin embargo en la fórmula más general de cálculo del valor esperado (que fue explicada en la sección 2.3) tanto las ganancias como las pérdidas se denominan “pagos”. Si el pago es positivo, se obtienen ganancias; si es negativo se obtienen pérdidas.

El algoritmo utilizado en el AnalizadorVE es una versión de ***Miximax*** (22), que aplica variantes de ésta fórmula para buscar la acción de mayor expectativa. *Miximax* es una forma de extender el algoritmo *Expectimax*[15] al póquer, ya que en éste juego no se puede hablar de valores exactos, si no de probabilidades, de manera que todas las acciones similares se agrupan en un único nodo que recibe una determinada probabilidad de que el oponente tenga un tipo u otro de cartas. Ésta probabilidad se basa en información conocida o estimada sobre el estado de la partida y el oponente. Debido a que elegir siempre la jugada de mayor valor esperado puede llevar a desarrollar un juego predecible se ha optado por combinar el uso de este analizador con los otros analizadores.

A continuación se muestra en pseudo-código dos de los métodos principales de la implementación llevada a cabo de dicho algoritmo.

[15] Algoritmo Minimax aplicado a dominios estocásticos.

```
dameJugadaMaximoVe(Nodo nodoActual, Nodo nodoInicial) {
	/*
	 * Creamos todas las opciones de hijos del nodo actual que no existan ya
	 */
	crearHijosDe(nodoActual);

	/*
	 * Calculamos el VE de cada posible jugada y las comparamos para averiguar
	 * la opción de mayor VE
	 */
	Nodo nodoMaximoVe = null;
	double veActual = Double.NEGATIVE_INFINITY;
	double veMaximo = veActual;

	for (int i = 0; i < nodoActual.dameNumHijos(); i++) {
		veActual = calculaVeDe(nodoActual.dameHijo(i), nodoInicial);
		if (veActual > veMaximo) {
			veMaximo = veActual;
			nodoMaximoVe = nodoActual.dameHijo(i);
		}
	}
	return nodoMaximoVe;
}

calculaVeDe(Nodo nodoActual, Nodo nodoInicial) {
	double ve;
	/* Abandonar */
	if (nodoActual.tipoAbandonar()) {
		ve = calculaVeDeAbandonar(nodoActual, nodoInicial);
	/* Pasar */
	} else if (nodoActual.tipoPasar()) {
		ve = calculaVeDePasar(nodoActual, nodoInicial);
	/* Igualar */
	} else if (nodoActual.tipoIgualar()) {
		ve = calculaVeDeIgualar(nodoActual, nodoInicial);
	/* Subir, contempla todos los tipos de subida */
	} else {
		ve = calculaVeDeSubir(nodoActual, nodoInicial);
	}

	return ve;
}
```

En cuanto a los oponentes, todos son representados por la clase **Oponente** que es un subtipo de la clase Jugador. Oponente contiene las acciones y datos que el programa

requiere de cada rival para poder aplicar su estrategia de juego. Su atributo más importante es de tipo Árbol, que almacena en su estructura la forma de jugar del oponente. En concreto cada oponente almacena una lista de árboles, ya que se crea un nuevo árbol por cada posición en la que éste ha jugado.

Tal y como se muestra en la Figura 11, cada uno de los nodos de la clase Arbol representa una acción en el juego y contiene información sobre:

- Nº de veces que se ha hecho dicha jugada
- El tipo de jugada que representa el nodo
- El jugador que efectuó la jugada
- Histograma de cartas mostradas por el rival en jugadas anteriores
- Tamaño medio del bote generado llegados a este punto
- Ganancias/pérdidas obtenidas (sólo en nodos hoja)
- Cartas del oponente (sólo en nodos hoja en que se mostraron)

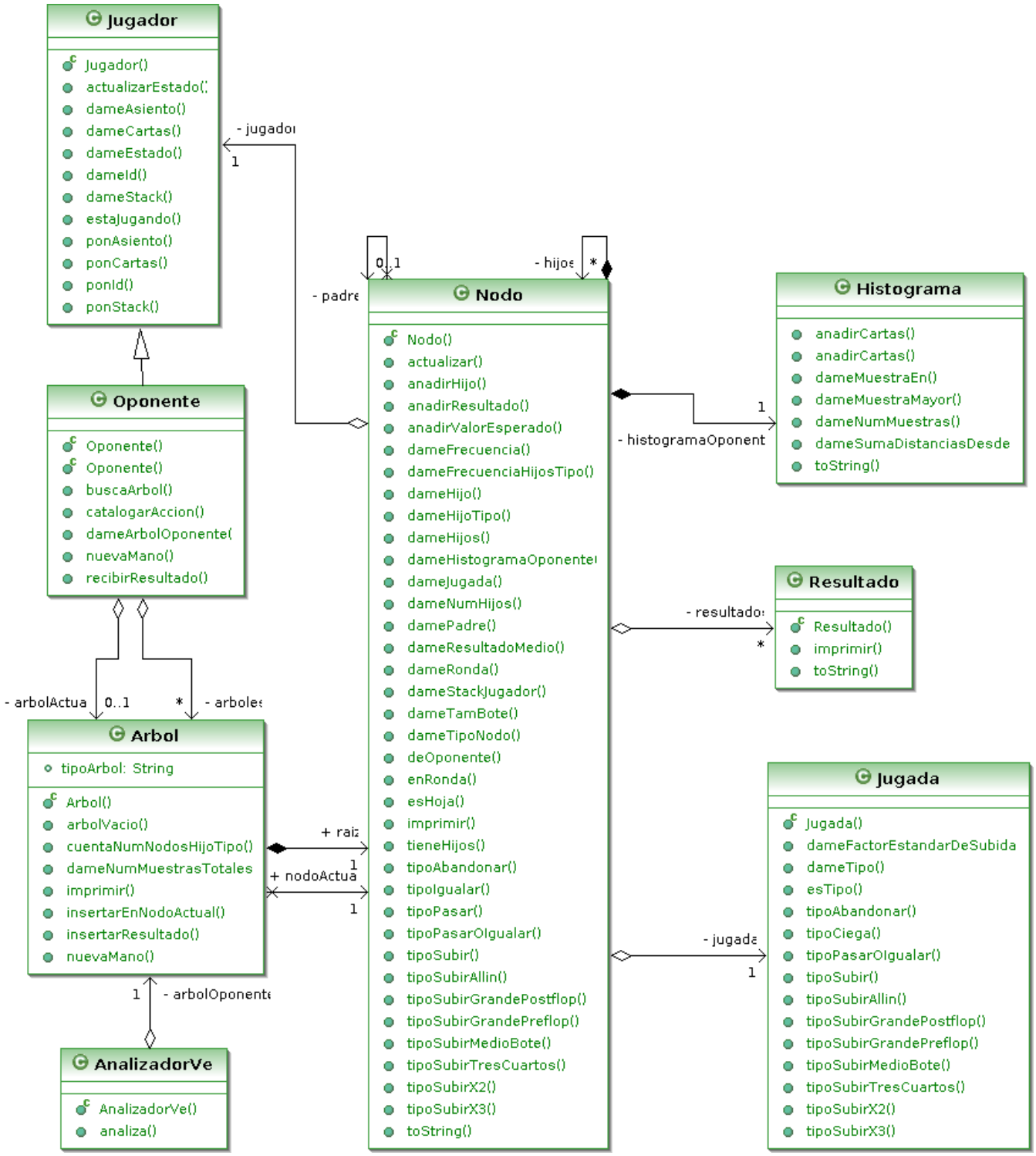



Figura 11: Diagrama de las clases que conforman el modelador de oponentes

## 3.4. Entorno

El entorno de ejecución de Uc3mBot está compuesto por un servidor de juego y la interfaz de usuario (en caso de querer crear una partida contra un humano) como se muestra en la Figura 12. En esta sección se explica cómo se añadieron estos componentes y sus características.

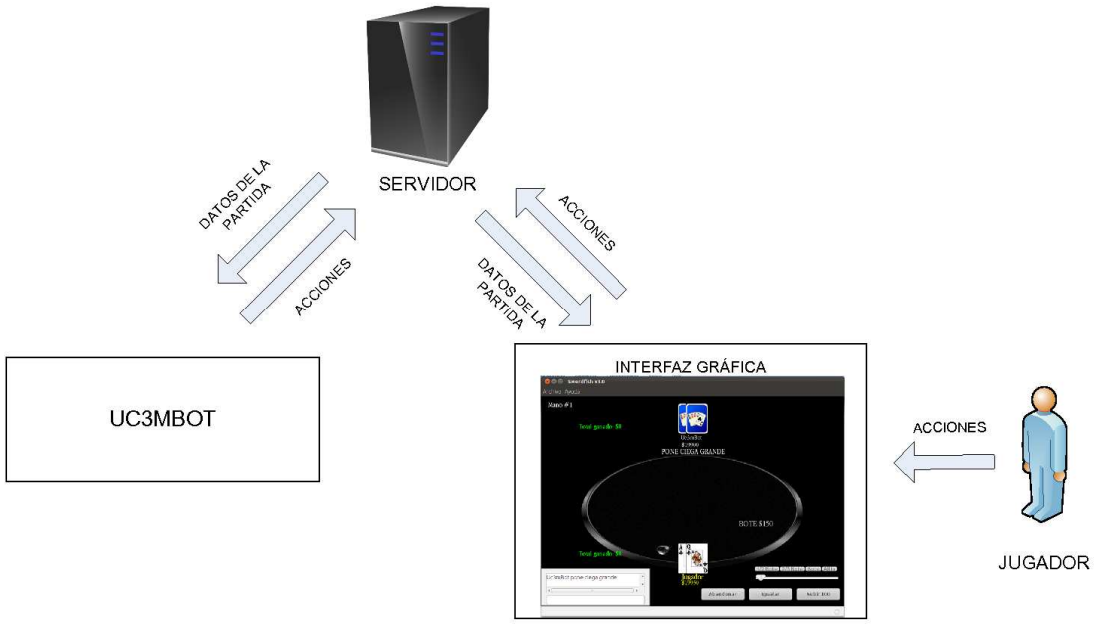



Figura 12: Entorno de ejecución de Uc3mBot

Por lo tanto, la aplicación se compone de 3 programas:

- El servidor de juego: Encargado de dirigir la partida, informando a todos los jugadores de cómo se desarrolla cada mano, así como de sus cartas y otra información relevante.
- El interfaz de juego: Utilizado para poder realizar partidas contra Uc3mBot.
- El jugador Uc3mBot: Explicado en las secciones 3.1, 3.2 y 3.3.

### 3.4.1. El servidor de juego

GlassFrog es el servidor de póquer utilizado, se trata de una distribución de la Annual Computer Poker Competition (ACPC), una competición entre agentes de póquer que se

realiza cada año durante la Asociación para el Avance de la Inteligencia Artificial (AAAI[16]) y que atrae competidores de todo el mundo. Dicha competición desarrolló su propio servidor de juego de forma que todos los participantes siguiesen un mismo protocolo de comunicación para el juego.

Dos versiones del protocolo han sido publicadas, la 1.0.0 y la 2.0.0, ambas difieren principalmente en pequeños aspectos prácticos. Nuestro agente ha sido diseñado para que pueda utilizar cualquiera de las dos versiones. Sin embargo la interfaz de usuario fue diseñada para utilizarse únicamente con la versión 1.0.0 del protocolo, como se explica en la sección 3.4.2.

Para utilizar un protocolo u otro basta con cambiar el tercer argumento que recibe el ejecutable de Uc3mBot poniendo un 1 para utilizar la versión 1.0.0 y un 2 para utilizar la versión más moderna, no es necesario modificar el código fuente.

La comunicación entre el servidor y cada uno de los jugadores se hace a través de TCP. Los jugadores (clientes) se conectan al servidor mediante su dirección IP y uno de los puertos ofrecidos por éste. El primer mensaje que debe enviar todo jugador es una cadena de caracteres indicando la versión del protocolo a utilizar. Por ejemplo, para utilizar la versión 2 se envía la cadena 'VERSION:2.0.0'.

Tras esto el servidor enviará repetidamente mensajes al cliente dando información sobre el estado del juego, incluyendo las manos en que el cliente no está jugando. Cada uno de estos mensajes tiene la forma:

'MATCHSTATE:' <posición> <nº de mano> <secuencia de apuestas> <cartas>

- El campo <posición> indica al jugador su posición en la mano relativa al repartidor. Un valor de 0 indica que el jugador es el primer jugador tras el repartidor, es decir, que es la ciega grande si la partida es de dos jugadores.
- <nº de mano> es simplemente un identificador de la mano en juego.
- El campo <secuencia de apuestas> indica la lista de acciones tomadas por los jugadores. El protocolo no hace distinciones entre igualar y pasar, pues

[16] Association for the Advancement of Artificial Intelligence.

representa ambas jugadas con la letra ‘c’; así como tampoco hace distinciones entre apostar o subir, representando ambas con la letra ‘r’ seguida de la cantidad apostada.

- <cartas> es el campo que representa todas las cartas que el jugador puede ver, es decir, sus dos cartas privadas mas las cartas comunitarias. En el caso de que al final de la mano uno de los rivales tenga que mostrar sus cartas, también se añadirán a este campo y por lo tanto serán visibles al jugador.

Finalmente añadir que cada jugador en su turno deberá contestar a los mensajes del servidor añadiéndoles la acción tomada al final de éstos. Un ejemplo de mensajes enviados y recibidos por el servidor con el protocolo 2.0.0 durante dos manos de una partida de dos jugadores sería:

*S-> MATCHSTATE:0:30::9s8h|*
*S-> MATCHSTATE:0:30:c:9s8h|*
*<-C MATCHSTATE:0:30:c:9s8h|:c*
*S-> MATCHSTATE:0:30:cc/:9s8h|/8c8d5c*
*<-C MATCHSTATE:0:30:cc/:9s8h|/8c8d5c:r250*
*S-> MATCHSTATE:0:30:cc/r250:9s8h|/8c8d5c*
*S-> MATCHSTATE:0:30:cc/r250c/:9s8h|/8c8d5c/6s*
*<-C MATCHSTATE:0:30:cc/r250c/:9s8h|/8c8d5c/6s:r500*
*S-> MATCHSTATE:0:30:cc/r250c/r500:9s8h|/8c8d5c/6s*
*S-> MATCHSTATE:0:30:cc/r250c/r500c/:9s8h|/8c8d5c/6s/2d*
*<-C MATCHSTATE:0:30:cc/r250c/r500c/:9s8h|/8c8d5c/6s/2d:r1250*
*S-> MATCHSTATE:0:30:cc/r250c/r500c/r1250:9s8h|/8c8d5c/6s/2d*
*S-> MATCHSTATE:0:30:cc/r250c/r500c/r1250c:9s8h|9c6h/8c8d5c/6s/2d*
*S-> MATCHSTATE:1:31::|JdTc*
*<-C MATCHSTATE:1:31::|JdTc:r300*
*S-> MATCHSTATE:1:31:r300:|JdTc*
*S-> MATCHSTATE:1:31:r300r900:|JdTc*
*<-C MATCHSTATE:1:31:r300r900:|JdTc:c*
*S-> MATCHSTATE:1:31:r300r900c/:|JdTc/6dJc9c*
*S-> MATCHSTATE:1:31:r300r900c/r1800:|JdTc/6dJc9c*
*<-C MATCHSTATE:1:31:r300r900c/r1800:|JdTc/6dJc9c:r3600*
*S-> MATCHSTATE:1:31:r300r900c/r1800r3600:|JdTc/6dJc9c*
*S-> MATCHSTATE:1:31:r300r900c/r1800r3600r9000:|JdTc/6dJc9c*
*<-C MATCHSTATE:1:31:r300r900c/r1800r3600r9000:|JdTc/6dJc9c:c*
*S-> MATCHSTATE:1:31:r300r900c/r1800r3600r9000c/:|JdTc/6dJc9c/Kh*
*S-> MATCHSTATE:1:31:r300r900c/r1800r3600r9000c/r20000:|JdTc/6dJc9c/Kh*
*<-C MATCHSTATE:1:31:r300r900c/r1800r3600r9000c/r20000:|JdTc/6dJc9c/Kh:c*
*S-> MATCHSTATE:1:31:r300r900c/r1800r3600r9000c/r20000c/:KsJs|JdTc/6dJc9c/Kh/Qc*

*Donde “S->” indica los mensajes que envía el servidor al cliente y “<-C” los que envía el cliente al servidor como respuesta.*

En cuanto al protocolo 1.0.0 se puede encontrar más información en el documento http://webdocs.cs.ualberta.ca/~pokert/code/nolimitprotocol.pdf.

### 3.4.2. La interfaz de usuario

La interfaz de usuario es una versión del proyecto Swordfish, proporcionado también por la ACPC. Su objetivo es permitir a jugadores humanos interactuar con agentes de póquer. Ésta interfaz fue desarrollada para utilizarse con el protocolo de comunicaciones en su versión 1.0.0 del servidor GlassFrog, por lo tanto Uc3mBot deberá configurarse para que utilice ese mismo protocolo en el caso de querer jugar una partida contra él. Para el presente proyecto se le han realizado una serie de modificaciones, como la traducción al español y la corrección de algunos errores, con el objetivo de hacerla más fácil e intuitiva de utilizar.

La interfaz consta de la mesa de juego, donde se muestra toda la información necesaria sobre el transcurso de cada una de las manos jugadas. Debajo del nombre de cada jugador se encuentra la cantidad de dinero o fichas por la que se está jugando la mano, dicha cantidad es reiniciada al principio de cada mano, pues son los marcadores situados en la parte izquierda de la ventana los que indican el total de ganancias (en verde) o pérdidas (en rojo) acumuladas. También se muestra en todo momento el tamaño del bote por el que se está jugando (Véase la Figura 13).


Figura 13: Interfaz de Usuario Swordfish traducida al español y mejorada

Para poder realizar las acciones de juego la interfaz de usuario provee de tres botones en la esquina inferior derecha de la ventana, que realizan las acciones Abandonar Igualar o Pasar y Subir una cantidad X.

Sobre estos tres botones principales se encuentran otra serie de botones de menor tamaño y una barra deslizadora cuya función es ayudar al usuario a escoger el tamaño de la apuesta.

Otro elemento importante es la consola de juego, que se muestra en la Figura 14. En ella se irán escribiendo automáticamente todas las acciones que han ido ocurriendo durante el transcurso de la partida, de modo que si el jugador desea hallar información sobre alguna mano anterior puede buscarla aquí.

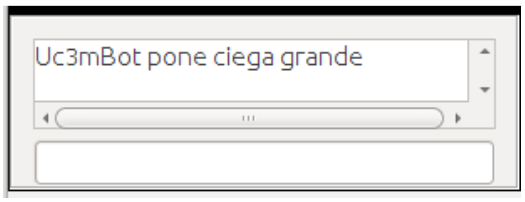



Figura 14: Consola de juego

Se puede salir de la partida en cualquier momento cerrando la ventana, pero si ya se ha realizado al menos una acción en el transcurso de la mano actual, el dinero apostado se quedará en el bote.

### 3.4.3. Librerías utilizadas

#### *XPokerEquity*

La mayoría del desarrollo se ha realizado en el lenguaje de programación Java, sin embargo el programa utiliza la librería "XPokerEquity" (desarrollada por James Devlin y descargable gratuitamente) cuyo código ha sido adaptado en este proyecto para poder ejecutarse en un entorno Linux. Ésta librería es la encargada de realizar gran parte de los cálculos referentes al valor esperado o esperanza matemática, y está desarrollada en C, puesto que este lenguaje generalmente permite obtener una mayor velocidad para este tipo de cálculos.

Se escogió esta librería frente a otras debido a su versatilidad, a que es uso libre y a su capacidad para calcular equidades frente a un rango completo de manos, superando a otras librerías de cálculo que sólo permiten calcular equidades frente a una mano concreta.

XPokerEquity es una calculadora de póquer capaz además de calcular la probabilidad de ganar o la equidad en cualquier situación que se produzca dentro de una mano de póquer, ya sea en la variante de Texas Hold'em (lo cual es ideal para nuestro propósito) como en la variante de Omaha. También permite realizar estos cálculos con cualquier número de jugadores que permita el número de cartas en la baraja. Por ejemplo, esta calculadora permite hacer cálculos del tipo:

Cartas comunitarias: 4♣ 5♦ A♣
651,429 partidas evaluadas a 25,517 partidas/segundo

| | equidad | %ganar | %empatar | botes ganados | botes empatados | rango de cartas |
|---|---|---|---|---|---|---|
| jugador 0: | 06.356% | **06.30%** | 00.05% | 41052 | 350.78 | { J♣J♥ } |
| jugador 1: | 14.329% | 13.34% | 00.99% | 86870 | 6474.53 | { 7♠8♠ } |
| jugador 2: | 09.735% | 09.20% | 00.54% | 59923 | 3492.28 | { 99+, AJs+ } |
| jugador 3: | 18.939% | 18.11% | 00.83% | 117976 | 5400.44 | { QQ+, AQs+, AQo+ } |
| jugador 4: | 10.087% | 08.64% | 01.45% | 56257 | 9455.03 | { random } |
| jugador 5: | 10.140% | 08.70% | 01.44% | 56675 | 9383.36 | { random } |
| jugador 6: | 10.080% | 08.63% | 01.45% | 56210 | 9457.86 | { random } |
| jugador 7: | 10.177% | 08.73% | 01.45% | 56874 | 9420.94 | { random } |
| jugador 8: | 10.157% | 08.71% | 01.45% | 56724 | 9447.28 | { random } |

Donde hay nueve jugadores en la partida con unas cartas comunitarias 4♣ 5♦ A♣, tenemos J♣J♥ y queremos saber nuestra probabilidad de ganar sabiendo que conocemos las cartas de otro jugador, de otros dos conocemos su rango de cartas y de los demás no tenemos información, luego pueden llevar cualquier par de cartas.

Ha sido necesario añadir código nuevo que actuase como interfaz a la hora de comunicar la parte de código escrita en Java con la librería XPokerEquity. Este código está escrito en el lenguaje Java Native Interface (JNI), que es una infraestructura digital de programación, que permite que un programa escrito en Java pueda interactuar con programas escritos en otros lenguajes como C, C++ y ensamblador. (Más información acerca de cómo programar en éste lenguaje puede encontrarse en (24) también descargable en internet desde http://java.sun.com/docs/books/jni/download/jni.pdf)

El interfaz escrito en lenguaje JNI está compuesto de los archivos "CalculadoraEquidad.c" y "CalculadoraEquidad.h", que se han creado de manera que permitan acceder a todas las rutinas necesarias de dichas librerías desde la clase "CalculadoraEquidad.java" del proyecto Uc3mBot.

La compilación de este código da como resultado la librería "libpokerjni.so" que forma parte de este proyecto y ha de estar situada en la misma carpeta en que esté el ejecutable del agente.

### *Poker-eval*

Además de la librería antes mencionada, se ha utilizado una versión en Java de la librería poker-eval, encargada ésta de evaluar el valor de cada mano de póquer. Para

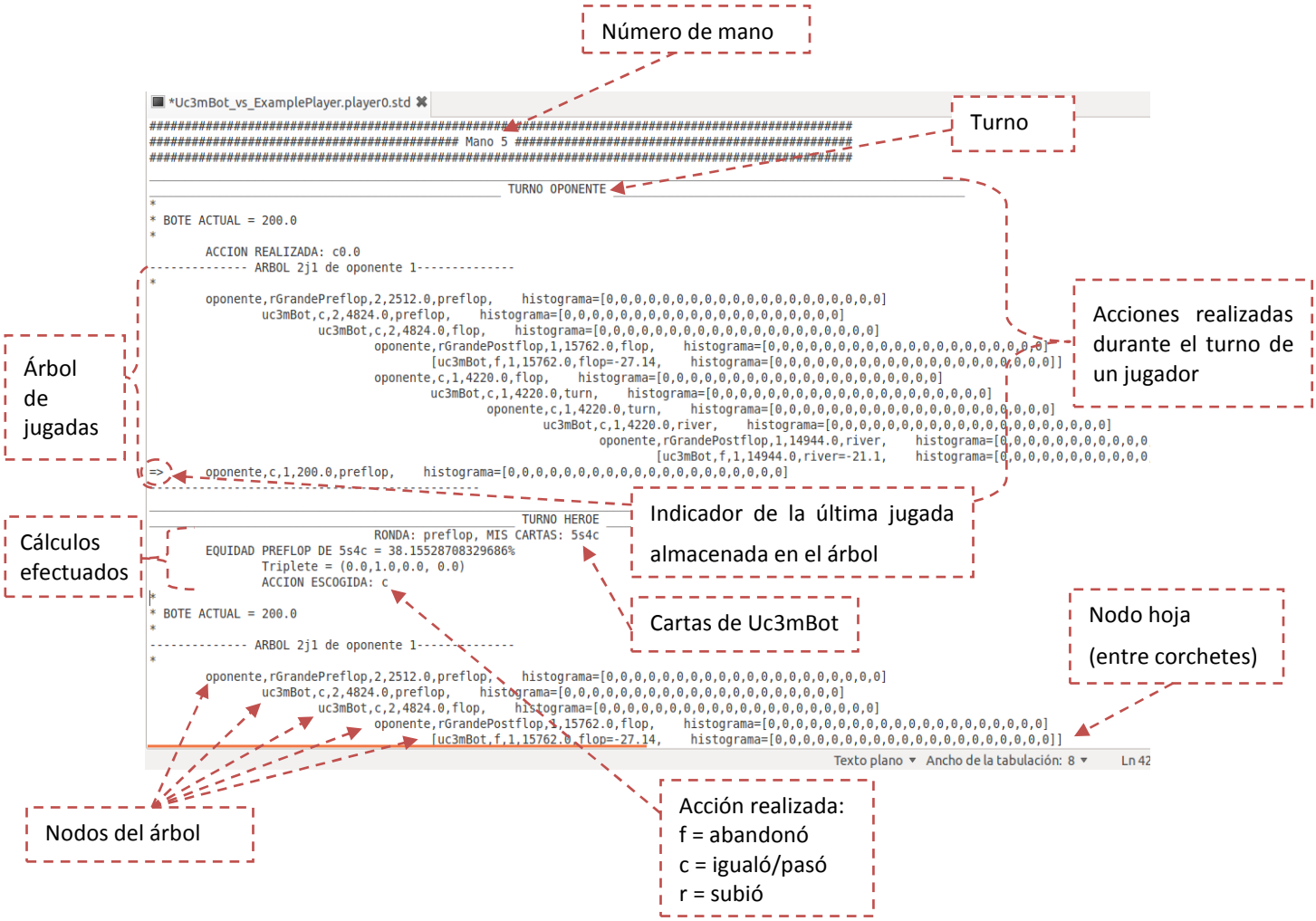
poder utilizarla se debe importar el archivo “poker-eval.jar” al proyecto. Ésta librería fue desarrollada a través del proyecto Pokersource de código abierto, del que se puede encontrar más información en http://gna.org/projects/pokersource/. Dicha librería se incluye dentro del proyecto. La clase que utiliza los métodos de ésta librería es “EvaluadorDeManos.java”.

## 3.5. Interpretación de los datos

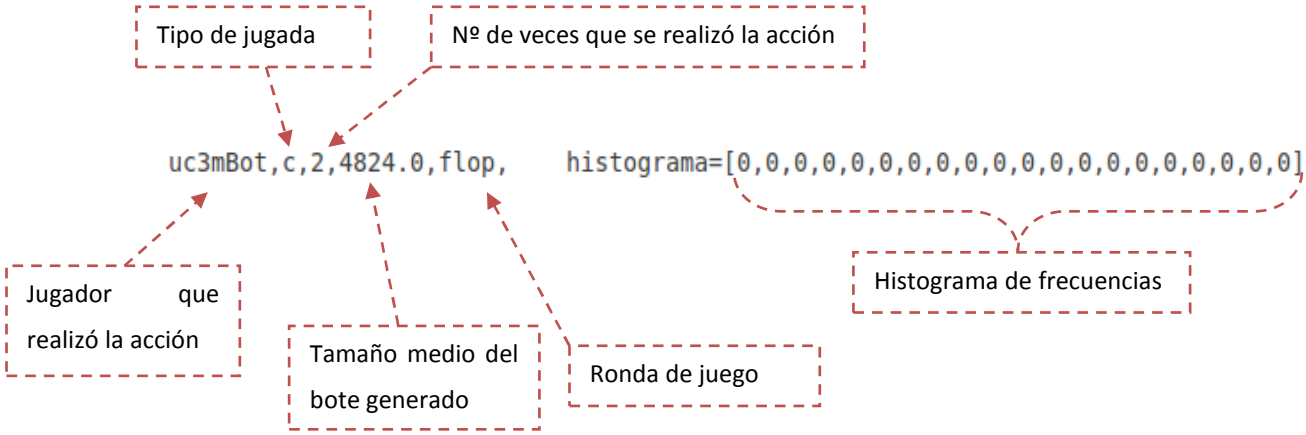
Puesto que Uc3mBot se conecta al servidor mediante un script, se ha redirigido la salida estándar al fichero executionLog.txt en la carpeta /bots del servidor, para así poder observar los mensajes que escribe durante su ejecución.

Durante la ejecución también se genera un archivo denominado historial.txt donde se imprimen los resultados de cada partida medidos en unidades de ciegas grandes. Es decir, si en la última mano Uc3mBot perdió 1600$ y la ciega grande vale 100$, en el historial se creará una línea con el valor -16, porque en esa mano se perdieron 16 ciegas grandes.

Todos los procesos y cálculos realizados por Uc3mBot durante una partida se imprimen en el fichero executionLog.txt mientras transcurre el juego. Dicho fichero se crea automáticamente en la carpeta donde esté alojado el ejecutable Uc3mBot.jar. A continuación se muestra un ejemplo de los datos que pueden encontrarse en dicho fichero y cómo se han de interpretar:




Cada uno de los nodos internos del árbol de jugadas contiene la siguiente información:



# Capítulo 4

## Pruebas y Resultados

El póquer no es un juego de azar como muchos jugadores creen, es un juego de habilidad. Que un jugador gane a otro una partida, no significa que uno sea mejor que el otro, pues en un juego donde el factor suerte es tan decisivo a corto plazo, no se puede medir la habilidad con exactitud.

Cuanto mayor sea el número de manos jugadas entre los jugadores, más fiables serán los resultados de juego, puesto que a la larga la suerte se igualará. De modo que el póquer es un juego cuyos resultados han de medirse a largo plazo, y esto es un hecho que todo jugador profesional de este juego conoce.

Cinco pruebas distintas han sido realizadas para probar el correcto funcionamiento de Uc3mBot, así como el cumplimiento de los objetivos propuestos al principio de este PFC. Todas las pruebas han consistido en medir la fortaleza de juego de Uc3mBot contra distintos oponentes a través de partidas de 25000 manos de duración (exceptuando la partida contra el jugador humano experto), que a pesar de que puedan parecer excesivamente largas, representan un mínimo de muestras para extraer conclusiones. Los resultados obtenidos se muestran a continuación.

## Primera prueba: Uc3mBot vs. SiempreIgualaBot

La primera prueba efectuada ha consistido en enfrentar a Uc3mBot contra un agente que únicamente realiza la acción de igualar. Este agente, bautizado con el nombre de SiempreIgualaBot, ha sido creado expresamente para probar que Uc3mBot se adapta bien a oponentes triviales.

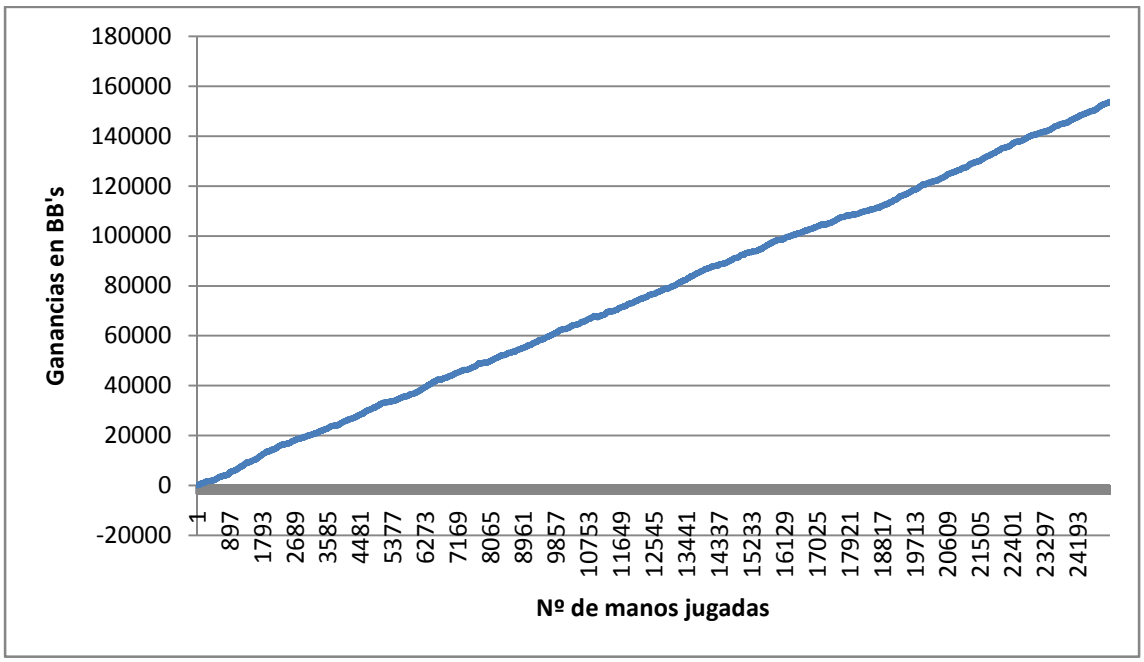



Gráfico 1: Resultados de la partida contra SiempreIgualaBot

Tal y como se puede observar en la gráfica, tras una partida de 25000 manos frente a SiempreIgualaBot, Uc3mBot demuestra una rápida adaptación al oponente desde las primeras manos, manteniendo su ratio de ganancias durante el enfrentamiento en una media de 6,28 BB's/mano, lo que indica que este tipo de rivales no suponen un obstáculo para nuestro agente.

## Segunda prueba: Uc3mBot vs. SiempreSubeBot

Para la segunda prueba se ha querido emplear un oponente similar al anterior en cuanto a su sencillez, pero que requiere un mayor nivel de adaptación para llegar a saber que sube con cualquieras dos cartas que le repartan.

Este oponente, denominado SiempreSubeBot, al igual que el rival de la prueba anterior, ha sido creado expresamente para este enfrentamiento.

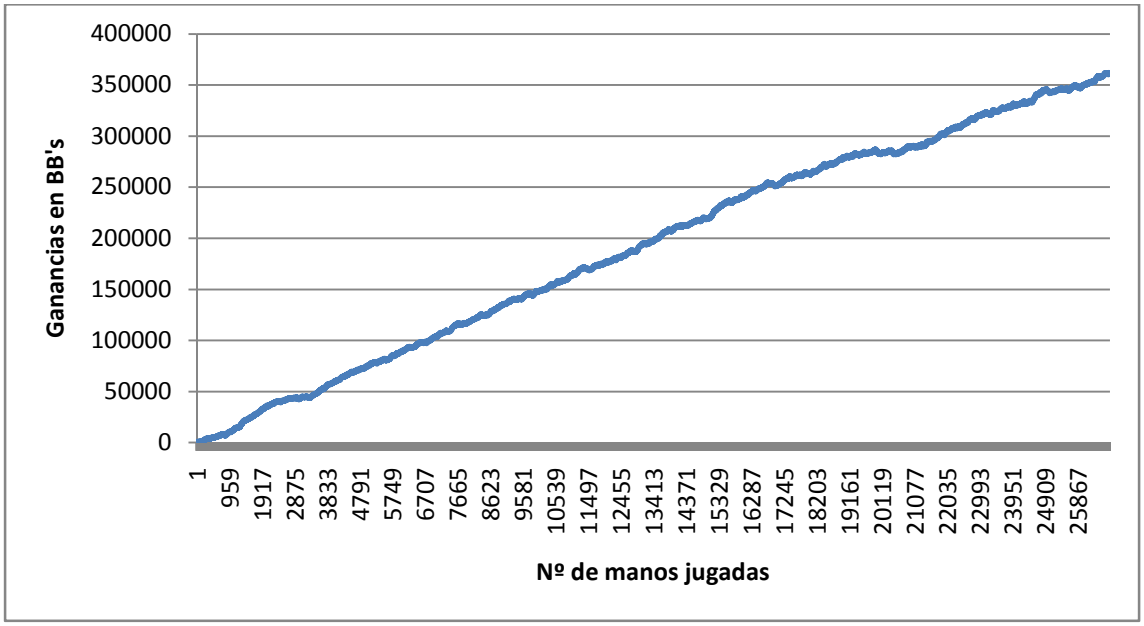



Gráfico 2: Resultados de la partida contra SiempreSubeBot

Como se puede observar por la gráfica, Uc3mBot también consigue batir a este oponente a un ritmo constante durante la partida jugada, de 25000 manos de duración, la misma cantidad que en la prueba anterior. Sin embargo en este caso sus ganancias aumentan mucho más rápido debido a que los botes por los que juega son mayores, llegando hasta los 361000 BB's de beneficios, lo que indica una ventaja imponente, de 14,4 BB's/mano jugada.

## Tercera prueba: Uc3mBot vs. Uc3mBotSimple

Se ha decidido efectuar la tercera prueba enfrentando a Uc3mBot contra la primera versión de él mismo, desarrollada al principio de este proyecto, a la que se ha decidido llamar Uc3mBotSimple.

Éste primer desarrollo, está basado principalmente en sistemas expertos para desarrollar su estrategia de juego, de modo que no utiliza ningún algoritmo de búsqueda al contrario que Uc3mBot, por lo que se puede medir la ventaja alcanzada mediante el uso de éste algoritmo.

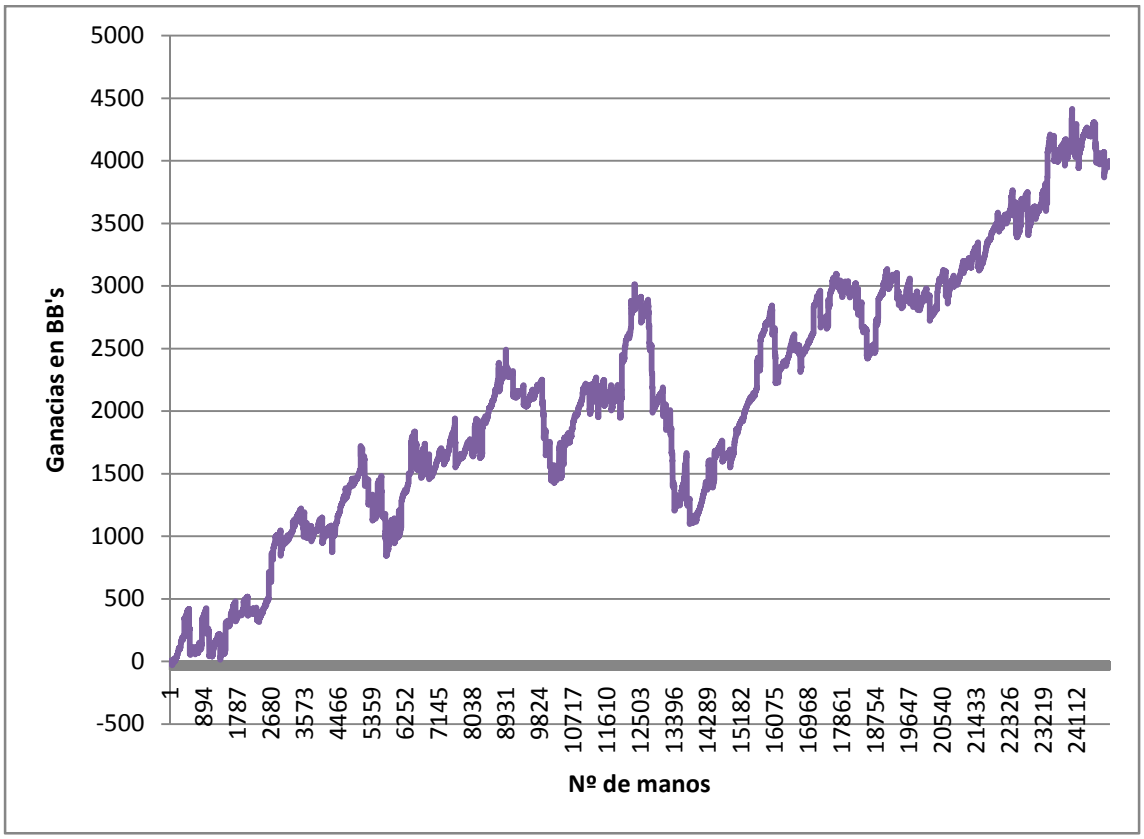



Gráfico 3: Resultados de la partida contra la primera versión de Uc3mBot

Tal y como puede verse en la gráfica, Uc3mBot fue también el ganador de esta tercera partida. Además se puede apreciar cómo al principio y mitad de la partida Uc3mBot está todavía adaptándose al juego del oponente, mostrando muchos altibajos en el juego, debido a que Uc3mBotSimple resulta mucho más impredecible que los dos adversarios anteriores.

En este caso el ratio de ganancias por mano jugada se acerca más a la normalidad, obteniendo unos beneficios de 0,16BB's/mano jugada.

## Cuarta prueba: Uc3mBot vs Hyperborean

Hyperborean es el agente actual campeón del mundo, proclamado en la última edición de la ACPC, en el año 2011, en la modalidad Texas Hold'em sin límite. Este jugador fue creado por la Computer Poker Research Group de la Universidad de Alberta (Canadá), a los que queremos agradecer enormemente primero, el habernos permitido probar nuestro agente contra Hyperborean, ya que consideramos que no habríamos podido

encontrar un adversario mejor, y segundo por toda la ayuda desinteresada que han mostrado a la hora de realizar esta prueba.

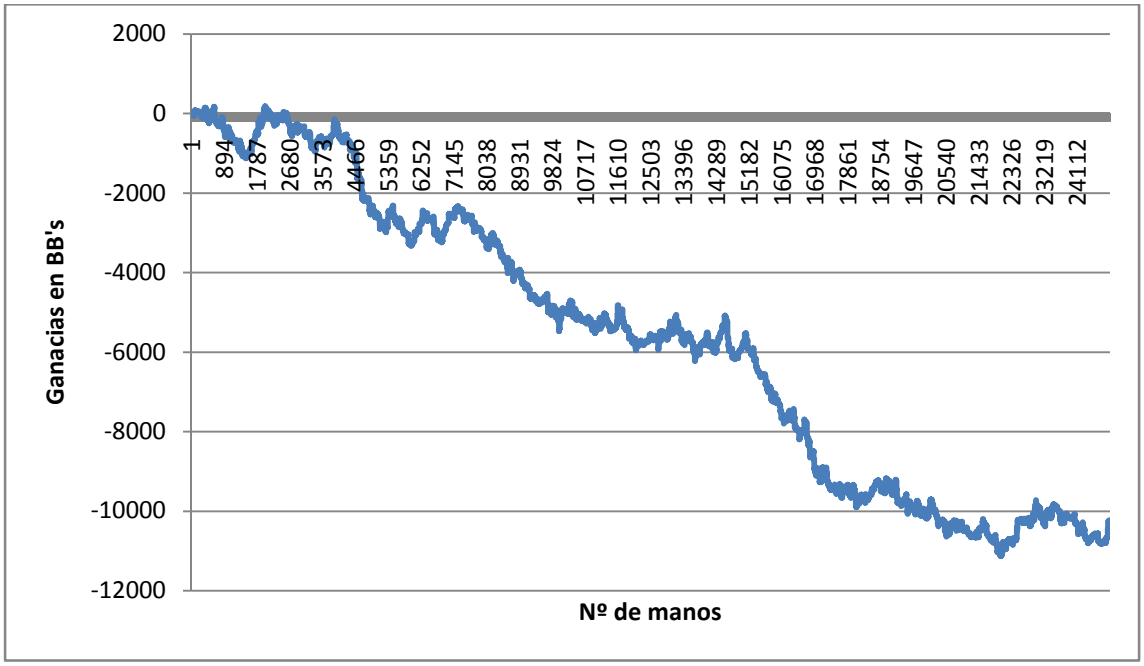



Gráfico 4: Resultados de la partida contra Hyperborean

Tal y como muestra la gráfica Hyperborean consigue obtener una ventaja clara en la partida, superando a nuestro agente, lo que en cierto modo era de esperar. Guiándonos por el Gráfico 4, parece que es al principio de la partida donde Uc3mBot muestra una oposición más dura para el rival, sin embargo conforme avanza el número de manos jugadas, Hyperborean consigue obtener una clara ventaja en los beneficios conseguidos.

En este enfrentamiento Uc3mBot pierde a un ritmo de -0,4BB’s/mano jugada, que es una cantidad menor que los resultados que obtuvieron la mayoría de oponentes en la ACPC del 2011 al enfrentarse a Hyperborean, donde de los cinco adversarios, sólo uno obtiene un mejor resultado que el obtenido aquí por nuestro agente, lo cual demuestra que está a la altura de presentarse al campeonato.

Pese a que ahora sabemos que no se va a conseguir un primer puesto en la competición, estos resultados demuestran que, a través de mejoras venideras, es

probable que nuestro agente consiga en un futuro quedar clasificado entre los primeros puestos.

## Comparativa de las partidas contra agentes

En la gráfica mostrada a continuación se ha realizado una comparativa de la evolución de las distintas partidas efectuadas contra otros agentes.

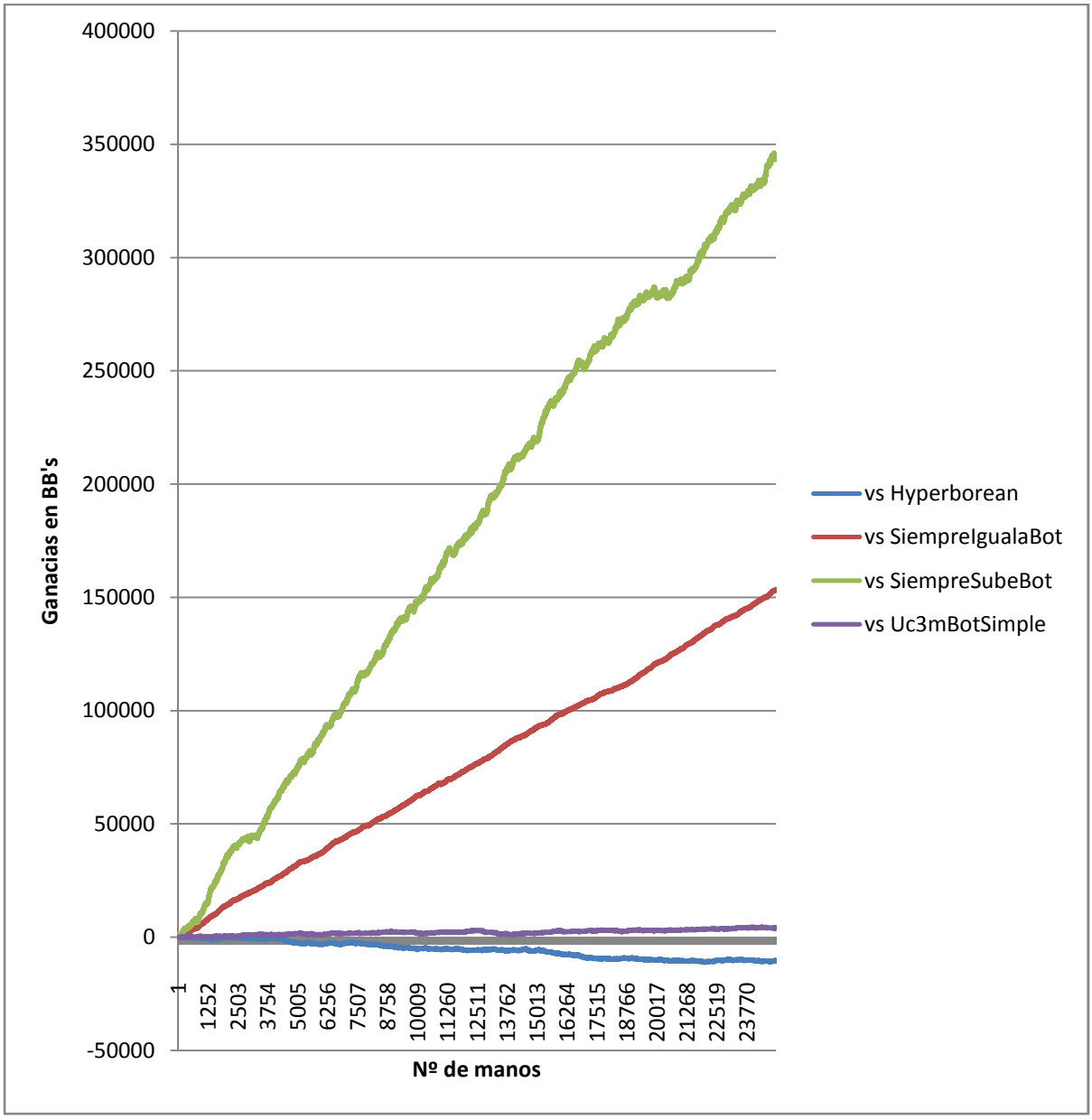



Gráfico 5: Evolución de las distintas partidas efectuadas

Como se puede observar, la diferencia de ganancias entre las distintas partidas es enorme, destacando el ratio de ganancias obtenido durante la partida contra SiempreSubeBot frente a las ganancias y pérdidas obtenidas contra Uc3mBotSimple y contra Hyperborean, que se asemejan mucho más a los resultados que se pudieran obtener entre oponentes de nivel parecido.

## Quinta prueba: Uc3mBot vs un jugador humano experto

En este caso sólo se jugaron alrededor de 500 manos de partida, por lo que resulta difícil extraer conclusiones de ella. Esto es debido a que los jugadores humanos resultan mucho más lentos a la hora de jugar y es difícil obtener una muestra significativa en un juego como el póquer cuando se enfrenta un agente contra un humano, ya que requeriría efectuar una partida del orden de varias semanas de duración, lo que en la mayoría de casos un jugador experto no está dispuesto a realizar.

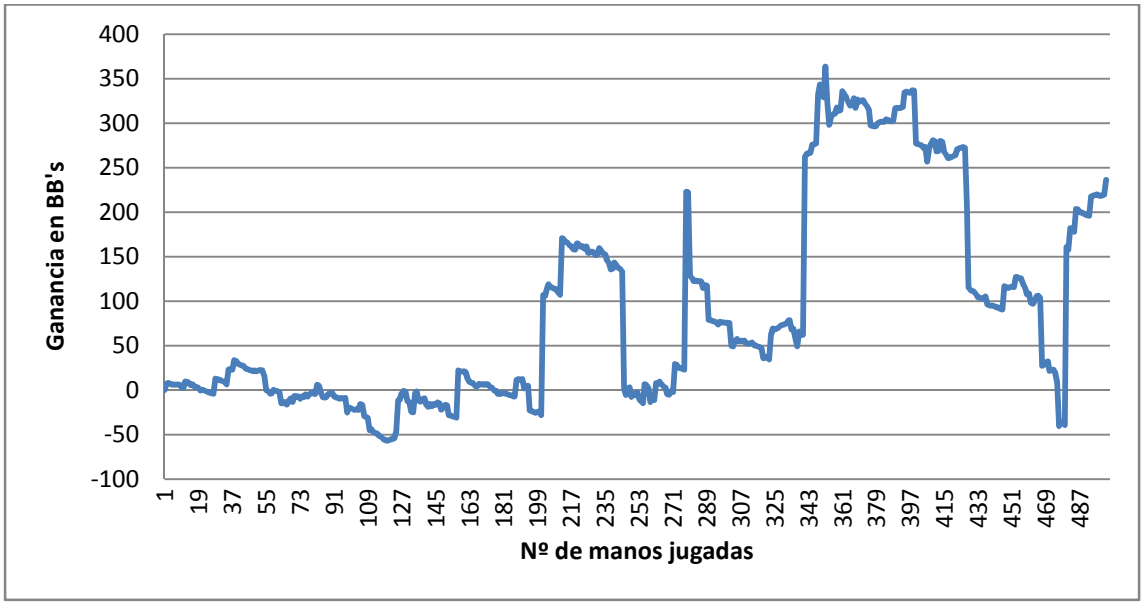



Gráfico 6: Resultados de la partida contra un jugador humano experto

Como puede observarse le es mucho más difícil a Uc3mBot adaptarse al juego de este jugador, ya que por lo general un humano resulta mucho más impredecible en el juego que un agente, esto lo demuestran las continuas fluctuaciones que representan las ganancias obtenidas durante la partida. Al final de la mano, Uc3mBot obtuvo una

pequeña ventaja frente al oponente humano, sin embargo predecir que Uc3mBot ganaría a la larga a un jugador humano experto resulta aventurarse demasiado, dada la pequeña muestra obtenida en esta prueba.

# Capítulo 5

# Conclusiones y Líneas Futuras

En este capítulo se exponen los comentarios del autor, recogiendo las impresiones recibidas y las conclusiones a las que ha llegado durante el desarrollo de este proyecto. También se incluyen algunas de las posibilidades que ofrece como líneas futuras de desarrollo, para la mejora y ampliación del agente.

## 5.1. Conclusiones

A lo largo del desarrollo de este PFC se han encontrado gran número de dificultades que han influido sobre el diseño inicial del agente, así como sobre la fase de implementación.

Este proyecto comenzó con la idea de crear un agente capaz de aprender de una gran cantidad de manos jugadas por el oponente y desarrollar una contraestrategia que superase a ése jugador. Sin embargo se ha ido transformado poco a poco en un proyecto más ambicioso a medida que se iba llevando a cabo, añadiendo más requisitos según se estimaba oportuno crear nuevas funcionalidades, hasta el punto que hubo que parar en algún punto, ya que uno podría estar mejorando su agente indefinidamente.

Muchas y distintas formas de estudiar el problema fueron estudiadas, probadas y descartadas, hasta dar con una solución que nos satisfizo. De hecho no fue hasta bien avanzado el proyecto que se decidió utilizar estructuras en forma de árbol para almacenar el modelo de cada oponente, y aplicar un algoritmo de búsqueda que buscase la jugada de mayor valor esperado sobre éste modelo.

Requirieron también mucho tiempo las pruebas que se realizaron con distintas librerías, hasta escoger las que se han presentado aquí como apoyo a los cálculos de Uc3mBot. A una de ellas se le tuvo que adaptar todo el código para que pudiese ejecutarse en un entorno Linux, lo que sumó una gran cantidad de tiempo al desarrollo. En cuanto a la otra librería, quiero expresar mis agradecimientos a los desarrolladores del proyecto Pokersource, por su inestimable ayuda en la adaptación de ésta a mi proyecto, ya que sin su colaboración el tiempo empleado hubiese sido mucho mayor.

Otras dificultades relacionadas con el entorno de ejecución surgieron al principio del desarrollo, concretamente una serie de errores que hacían que la Interfaz de Usuario no funcionase correctamente y que hubo que corregir rápidamente.

También fue difícil en la fase final del proyecto adaptar el jugador a los requisitos de velocidad de juego necesarios para poder efectuar partidas contra otros agentes, los cuales se mostraban mucho más rápidos en la toma de decisiones. No obstante se consiguió aumentar la velocidad media de juego de 150 manos/hora a más de 5000 manos/hora, a través de la mejora de algunos aspectos del algoritmo Miximax utilizado, y sacrificando muy poca precisión de cálculo, lo que no deja de ser un logro.

A pesar de todos los buenos resultados obtenidos durante las pruebas, tengo la sensación de que la parte divertida, por así decirlo, acaba de comenzar, pues lo más difícil de este proyecto ya está completado, que es todo el diseño del agente, y ahora es posible realizar pequeñas modificaciones o añadir nuevas funcionalidades que no requieren mucho tiempo, y pasar rápidamente a estudiar los distintos progresos que el agente muestra en el juego.

## 5.2. Líneas futuras

Pese a lo extenso del trabajo realizado, éste no deja de considerarse como un primer paso hacia el desarrollo de futuros agentes, facilitando con este modelo la creación de nuevos avances y futuras mejoras. Por lo tanto son muchas las líneas de trabajo que pueden tomarse a partir de aquí. A continuación se exponen algunas de ellas.

La primera mejora propuesta consistiría en eliminar progresivamente las muestras antiguas del juego del rival, para que el programa se adapte más rápidamente a los cambios en el estilo de juego del oponente. La mejor forma de hacerlo sería a través de un algoritmo que asignase unos pesos a cada muestra, de manera que las últimas muestras tuviesen más protagonismo en el cálculo del VE. Así, según las muestras fuesen más antiguas, éstas deberían tenerse menos en cuenta.

Otra mejora significativa consistiría en expandir los cálculos llevados a cabo por el programa en los nodos en que influye la suerte. Pese a que con ello se sacrificaría cierta velocidad de juego, se terminaría ganando en precisión a la hora de calcular el valor esperado de cada jugada. Una posible forma de no perder mucha velocidad de juego tras aplicar estos cambios, podría ser clasificando todas las combinaciones de cartas que pueden salir en tres grupos distintos, las favorables, las inocuas y las desfavorables. De este modo el número de estados de juego o nodos del árbol a evaluar disminuiría considerablemente sin sacrificar demasiada precisión de cálculo.

Una tercera proposición consistiría en adaptar Uc3mBot para que juegue en partidas de más de 2 jugadores, lo que supone un nuevo reto para el proyecto. En este caso el agente debería lidiar con un número mucho mayor de estados de juego, proporcional a la cantidad de oponentes en la mesa.

Finalmente es necesario señalar que es muy importante que el agente consiga tomar las decisiones de juego en el mínimo tiempo posible, con el objetivo de superar las restricciones de tiempo de juego aplicadas en la competición de la ACPC. Por este motivo se deben agilizar los cálculos al máximo tras cualquier mejora aplicada, para así

aumentar la velocidad del agente en la toma de decisiones y cumplir con este requisito.

# Bibliografía


1. **Johanson, Michael Bradley.** *Robust Strategies and Counter-Strategies: Building a Champion Level Computer.* 2007. págs. 79-90.

2. **Neumann, John von y Morgenstern, Oskar.** *Theory of Games and Economic Behavior.* 1944. págs. 186-218. ISBN 978-0-691-13061-3.

3. **Nash, John F. and Shapley, L. S.** *A simple three person poker game. Contributions to the theory of games.* 1950. pp. 105-116. Vol. 1.

4. **Nash, John F.** *Non-cooperative games. Annals of Mathematics.* 1951. pp. 286-295. Vol. 54.

5. **Cutler, William H.** *An Optimal Strategy for Pot-Limit Poker.* 1975. págs. 368-376.

6. **Zadeh, Norman.** *Winning Poker Systems.* 1974.

7. **Ankeny, Nesmith C.** *Poker Strategy: Winning with Game Theory.* 1981.

8. **Findler, N. V.** *Computer Model of Gambling and Bluffing.* 1961. págs. 5-6.

9. —. *Some New Approaches to Machine Learning.* 1969. págs. 173-182.

10. **Findler, N. V., y otros.** *Studies on Decision Making Using the Game of Poker.* 1972.

11. **Findler, N. V., Klein, H. y Levine, Z.** *Experiments with inductive discovery processes leading to heuristics in a poker program.* Berlin : s.n., 1973.

12. **Findler, N. V.** *Studies in Machine Cognition Using the Game of Poker.* 1977.

13. —. *Computer Poker.* s.l. : Scientific American, 1978.

14. **Schaeffer, Jonathan y Lake, Robert.** *Solving the Game of Checkers.* 1996.

15. **Billings, Darse.** *Computer Poker.* 1995.

16. **Papp, Denis Richard.** *Dealing with Imperfect Information in Poker.* Edmonton, Alberta : s.n., 1998.

17. **Darse Billings, Denis Papp, Lourdes Peña, Jonathan Schaeffer, Duane Szafron.** *Using Selective-Sampling Simulations in Poker.* 1999.

18. **Jonathan Schaeffer, Darse Billings, Lourdes Peña, Duane Szafron.** *Learning to Play Strong Poker.* 1999.

19. **Darse Billings, Lourdes Pena, Jonathan Schaeffer, Duane Szafron.** *Using Probabilistic Knowledge and Simulation to play Poker.* 1999.

20. **Davidson, Aaron.** *Opponent Modeling in Poker: Learning and Acting in a Hostile and Uncertain Environment.* 2002.

21. **Aaron Davidson, Darse Billings, Jonathan Schaeffer, and Duane Szafron.** *Improved Opponent Modeling in Poker.* 2000.

22. **Darse Billings, Aaron Davidson, Terence Schauenberg, Neil Burch, Michael Bowling, Robert Holte, Jonathan Schaeffer, and Duane Szafron.** *Game tree search with adaptation in stochastic imperfect information games.* 2004.

23. **poker, State translation in no-limit.** *David Paul Schnizlein.* 2009.

24. **Liang, Sheng.** *Java Native Interface: Programmer's Guide and Specification.* s.l. : Addison-Wesley, 1999. ISBN.

25. **Sklansky, David.** *The Theory of Poker.* s.l. : Two plus two publishing, 1992.

26. **Papp, Denis.** *Dealing with Imperfect Information in Poker.* 1998.

27. **Darse Billings, Denis Papp, Jonathan Schaeffer, and Duane Szafron.** *Poker as a Testbed for Machine Intelligence Research.* 1998.

28. **Darse Billings, Aaron Davidson, Jonathan Schaeffer, and Duane Szafron.** *The Challenge Of Poker.* 2002.

29. **Newman, Donald J.** *A model for 'real' poker.* 1959.

# Anexo A.

## Planificación y presupuesto

En este documento se describe cómo se ha planificado el proyecto desde las primeras etapas hasta su finalización. También se expone el cálculo del presupuesto que conlleva su desarrollo.

## Planificación

Dada la magnitud de este proyecto, ha sido necesario realizar una planificación para distribuir la cantidad de tiempo previsto en las diferentes tareas o actividades efectuadas durante su desarrollo.

A este efecto se ha utilizado un diagrama de Gantt (como se muestra en Figura 15 y Figura 16), que se ha convertido en una herramienta básica en la gestión de proyectos de todo tipo. En este diagrama se muestran el origen y final de las diferentes unidades mínimas de trabajo y de los grupos de tareas.

Tal y como se puede observar se han incluido también las reuniones con el tutor y las tareas de documentación, que inicialmente sirvieron para adquirir los conocimientos necesarios sobre las diferentes técnicas de desarrollo de agentes de póquer; y posteriormente se emplearon en realizar un análisis en mayor profundidad para la redacción del estado del arte. Por último podemos destacar que, de entre las tareas principales, la de implementación es con mucho la que más tiempo han consumido.

| Id | Modo de tarea | Nombre de tarea | Duración | Comienzo | Fin |
|---|---|---|---|---|---|
| 1 | | Inicio Proyecto | 0 días | lun 07/03/11 | lun 07/03/11 |
| 2 | | **Reuniones con el tutor** | **355 días** | **lun 07/03/11** | **vie 13/07/12** |
| 3 | | Reunión 1 | 1 día | lun 07/03/11 | lun 07/03/11 |
| 4 | | Reunión 2 | 1 día | lun 14/03/11 | lun 14/03/11 |
| 5 | | Reunión 3 | 1 día | jue 14/04/11 | jue 14/04/11 |
| 6 | | Reunión 4 | 1 día | jue 19/05/11 | jue 19/05/11 |
| 7 | | Reunión 5 | 1 día | lun 06/06/11 | lun 06/06/11 |
| 8 | | Reunión 6 | 1 día | lun 10/10/11 | lun 10/10/11 |
| 9 | | Reunión 7 | 1 día | mié 02/11/11 | mié 02/11/11 |
| 10 | | Reunión 8 | 1 día | jue 01/12/11 | jue 01/12/11 |
| 11 | | Reunión 9 | 1 día | lun 16/01/12 | lun 16/01/12 |
| 12 | | Reunión 10 | 1 día | jue 02/02/12 | jue 02/02/12 |
| 13 | | Reunión 11 | 1 día | lun 26/03/12 | lun 26/03/12 |
| 14 | | Reunión 12 | 1 día | vie 27/04/12 | vie 27/04/12 |
| 15 | | Reunión 13 | 1 día | mié 06/06/12 | mié 06/06/12 |
| 16 | | Reunión 14 | 1 día | lun 11/06/12 | lun 11/06/12 |
| 17 | | Reunión 15 | 1 día | vie 29/06/12 | vie 29/06/12 |
| 18 | | Reunión 16 | 1 día | lun 09/07/12 | lun 09/07/12 |
| 19 | | Reunión 17 | 1 día | mié 11/07/12 | mié 11/07/12 |
| 20 | | Reunión 18 | 1 día | vie 13/07/12 | vie 13/07/12 |
| 21 | | **Análisis** | **45 días** | **mar 08/03/11** | **lun 09/05/11** |
| 22 | | Propuesta del proyecto | 1 día | mar 08/03/11 | mar 08/03/11 |
| 23 | | Estudio preliminar | 35 días | mié 09/03/11 | mar 26/04/11 |
| 24 | | Estudio de librerías necesarias | 9 días | mié 27/04/11 | lun 09/05/11 |
| 25 | | Fin fase análisis | 0 días | lun 09/05/11 | lun 09/05/11 |
| 26 | | **Diseño** | **55 días** | **mar 10/05/11** | **lun 25/07/11** |
| 27 | | Diseño del entorno | 15 días | mar 10/05/11 | lun 30/05/11 |
| 28 | | Arquitectura del agente | 20 días | mar 31/05/11 | lun 27/06/11 |
| 29 | | Diagrama de clases | 20 días | mar 28/06/11 | lun 25/07/11 |
| 30 | | Fin fase diseño | 0 días | lun 25/07/11 | lun 25/07/11 |
| 31 | | **Implementación** | **135 días** | **mar 26/07/11** | **lun 30/01/12** |
| 32 | | Interfaz de conexión | 12 días | mar 26/07/11 | mié 10/08/11 |
| 33 | | Modelo del mundo | 24 días | jue 11/08/11 | mar 13/09/11 |
| 34 | | Adaptación de librerías | 10 días | mié 14/09/11 | mar 27/09/11 |
| 35 | | Modelador de oponentes | 35 días | mié 28/09/11 | mar 15/11/11 |
| 36 | | Analizador preflop | 13 días | mié 28/09/11 | vie 14/10/11 |
| 37 | | Analizador postflop | 21 días | lun 17/10/11 | lun 14/11/11 |
| 38 | | Analizador de valor esperado | 54 días | mié 16/11/11 | lun 30/01/12 |
| 39 | | Fin fase implementación | 0 días | lun 30/01/12 | lun 30/01/12 |
| 40 | | **Pruebas** | **25 días** | **mar 31/01/12** | **lun 05/03/12** |
| 41 | | Creación de agentes de pru | 4 días | mar 31/01/12 | vie 03/02/12 |
| 42 | | Realización de partidas | 18 días | lun 06/02/12 | mié 29/02/12 |
| 43 | | Análisis de resultados | 3 días | jue 01/03/12 | lun 05/03/12 |
| 44 | | Fin fase pruebas | 0 días | lun 05/03/12 | lun 05/03/12 |
| 45 | | Documentación | 349 días | mar 15/03/11 | vie 13/07/12 |
| 46 | | Fin proyecto | 0 días | vie 13/07/12 | vie 13/07/12 |

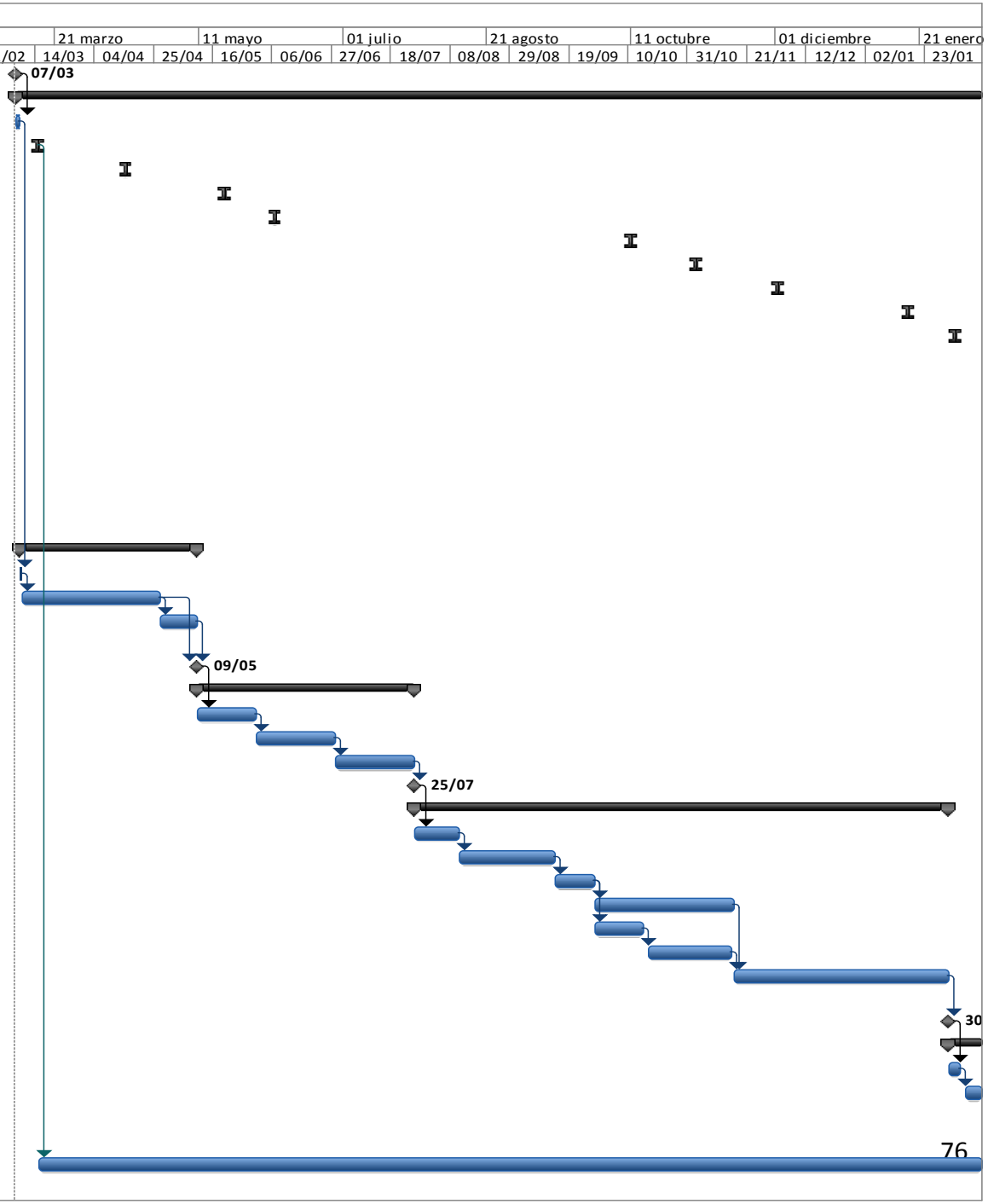



Figura 15: Diagrama de Gantt (parte 1)

| Id | Modo de tarea | Nombre de tarea | Duración | Comienzo | Fin |
|---|---|---|---|---|---|
| 1 | | Inicio Proyecto | 0 días | lun 07/03/11 | lun 07/03/11 |
| 2 | | **Reuniones con el tutor** | **355 días** | **lun 07/03/11** | **vie 13/07/12** |
| 3 | | Reunión 1 | 1 día | lun 07/03/11 | lun 07/03/11 |
| 4 | | Reunión 2 | 1 día | lun 14/03/11 | lun 14/03/11 |
| 5 | | Reunión 3 | 1 día | jue 14/04/11 | jue 14/04/11 |
| 6 | | Reunión 4 | 1 día | jue 19/05/11 | jue 19/05/11 |
| 7 | | Reunión 5 | 1 día | lun 06/06/11 | lun 06/06/11 |
| 8 | | Reunión 6 | 1 día | lun 10/10/11 | lun 10/10/11 |
| 9 | | Reunión 7 | 1 día | mié 02/11/11 | mié 02/11/11 |
| 10 | | Reunión 8 | 1 día | jue 01/12/11 | jue 01/12/11 |
| 11 | | Reunión 9 | 1 día | lun 16/01/12 | lun 16/01/12 |
| 12 | | Reunión 10 | 1 día | jue 02/02/12 | jue 02/02/12 |
| 13 | | Reunión 11 | 1 día | lun 26/03/12 | lun 26/03/12 |
| 14 | | Reunión 12 | 1 día | vie 27/04/12 | vie 27/04/12 |
| 15 | | Reunión 13 | 1 día | mié 06/06/12 | mié 06/06/12 |
| 16 | | Reunión 14 | 1 día | lun 11/06/12 | lun 11/06/12 |
| 17 | | Reunión 15 | 1 día | vie 29/06/12 | vie 29/06/12 |
| 18 | | Reunión 16 | 1 día | lun 09/07/12 | lun 09/07/12 |
| 19 | | Reunión 17 | 1 día | mié 11/07/12 | mié 11/07/12 |
| 20 | | Reunión 18 | 1 día | vie 13/07/12 | vie 13/07/12 |
| 21 | | **Análisis** | **45 días** | **mar 08/03/11** | **lun 09/05/11** |
| 22 | | Propuesta del proyecto | 1 día | mar 08/03/11 | mar 08/03/11 |
| 23 | | Estudio preliminar | 35 días | mié 09/03/11 | mar 26/04/11 |
| 24 | | Estudio de librerías necesarias | 9 días | mié 27/04/11 | lun 09/05/11 |
| 25 | | Fin fase análisis | 0 días | lun 09/05/11 | lun 09/05/11 |
| 26 | | **Diseño** | **55 días** | **mar 10/05/11** | **lun 25/07/11** |
| 27 | | Diseño del entorno | 15 días | mar 10/05/11 | lun 30/05/11 |
| 28 | | Arquitectura del agente | 20 días | mar 31/05/11 | lun 27/06/11 |
| 29 | | Diagrama de clases | 20 días | mar 28/06/11 | lun 25/07/11 |
| 30 | | Fin fase diseño | 0 días | lun 25/07/11 | lun 25/07/11 |
| 31 | | **Implementación** | **135 días** | **mar 26/07/11** | **lun 30/01/12** |
| 32 | | Interfaz de conexión | 12 días | mar 26/07/11 | mié 10/08/11 |
| 33 | | Modelo del mundo | 24 días | jue 11/08/11 | mar 13/09/11 |
| 34 | | Adaptación de librerías | 10 días | mié 14/09/11 | mar 27/09/11 |
| 35 | | Modelador de oponentes | 35 días | mié 28/09/11 | mar 15/11/11 |
| 36 | | Analizador preflop | 13 días | mié 28/09/11 | vie 14/10/11 |
| 37 | | Analizador postflop | 21 días | lun 17/10/11 | lun 14/11/11 |
| 38 | | Analizador de valor esperado | 54 días | mié 16/11/11 | lun 30/01/12 |
| 39 | | Fin fase implementación | 0 días | lun 30/01/12 | lun 30/01/12 |
| 40 | | **Pruebas** | **25 días** | **mar 31/01/12** | **lun 05/03/12** |
| 41 | | Creación de agentes de pru | 4 días | mar 31/01/12 | vie 03/02/12 |
| 42 | | Realización de partidas | 18 días | lun 06/02/12 | mié 29/02/12 |
| 43 | | Análisis de resultados | 3 días | jue 01/03/12 | lun 05/03/12 |
| 44 | | Fin fase pruebas | 0 días | lun 05/03/12 | lun 05/03/12 |
| 45 | | Documentación | 349 días | mar 15/03/11 | vie 13/07/12 |
| 46 | | Fin proyecto | 0 días | vie 13/07/12 | vie 13/07/12 |

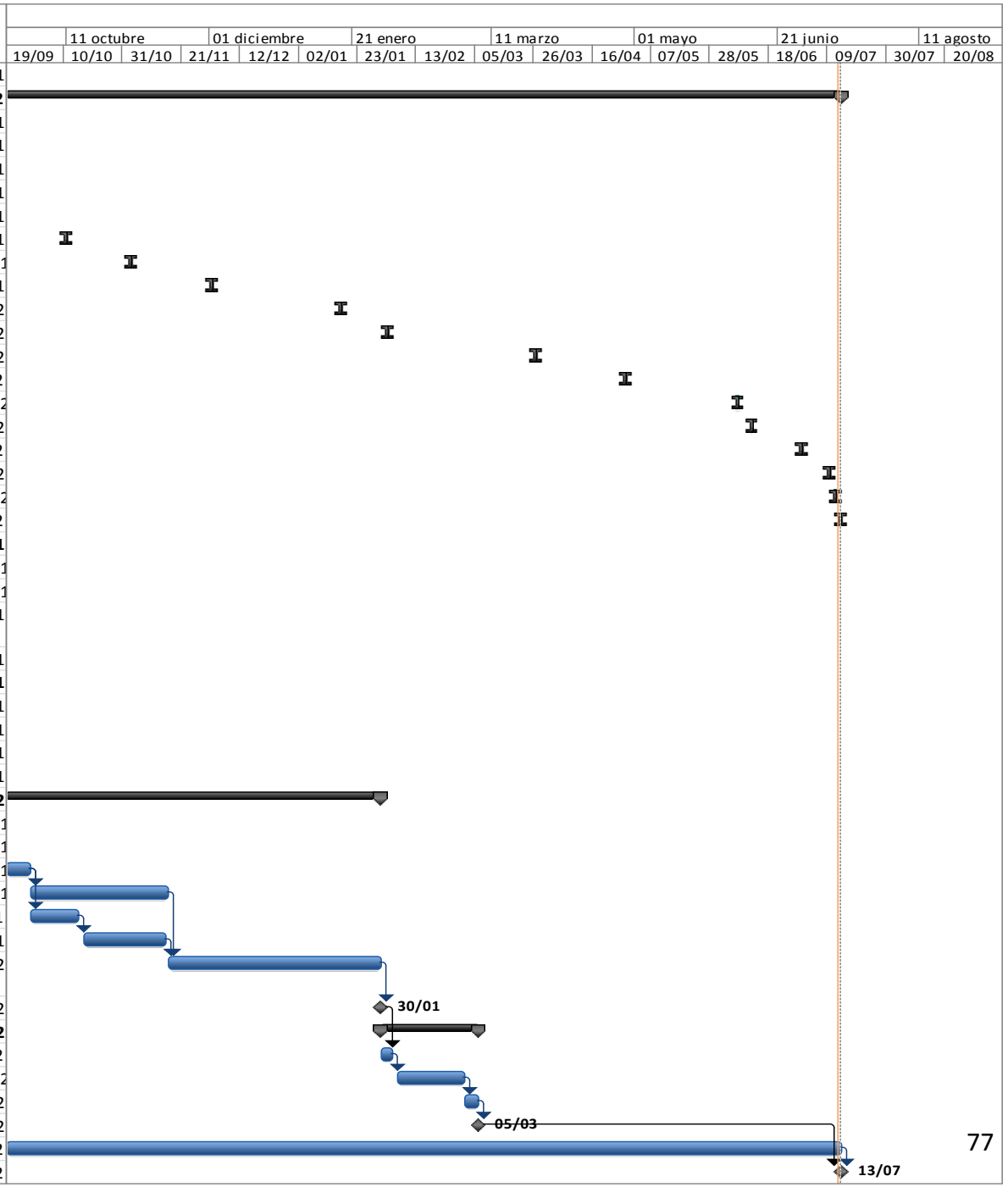





Figura 16: Diagrama de Gantt (parte 2)

# Presupuesto

En este apartado se detalla el presupuesto del proyecto, especificando los gastos de personal, hardware y software en que se ha incurrido durante el desarrollo.

Se ha planificado un calendario laboral compuesto por jornadas de tres horas por día. Pese a que la línea de trabajo ha sido irregular (en los periodos no lectivos las jornadas fueron de muchas horas más de trabajo, pero fueron compensadas con otras de menos trabajo), se ha supuesto un esfuerzo continuo a lo largo de todo el proceso, con lo que finalmente la solución tomada refleja una media real de las horas invertidas en el desarrollo del proyecto.

Las horas de trabajo invertidas en este proyecto han sido distribuidas de la siguiente forma entre las distintas fases de su desarrollo:

- Análisis: 45 días a 3 horas/día = 135 horas
- Diseño: 55 días a 3 horas/día = 165 horas
- Implementación: 135 días a 4 horas/día = 540 horas
- Pruebas: 25 días a 2 horas/día = 50 horas
- Documentación: 122 días * 2 horas/día = 244 horas

El total de tiempo invertido en el proyecto asciende por lo tanto a 1134 horas de trabajo.

Los siguientes apartados son el desglose de la plantilla presupuestaria proporcionada por la Universidad Carlos III de Madrid donde se estima una dedicación por empleado de 131,25 horas/mes.

## Coste de personal

El coste de personal empleado en este proyecto se detalla en la tabla mostrada a continuación

| Profesional | Horas | Coste(€)/hora | Total(€) |
|---|---|---|---|
| **Analista** | 135 | 33 | 4455 |
| **Diseñador** | 165 | 33 | 5445 |
| **Programador** | 540 | 25 | 13500 |
| **Responsable de pruebas** | 50 | 15 | 750 |
| **Responsable de documentación** | 244 | 15 | 3660 |
| | | | **27810 €** |

Tabla 2: Coste de personal

## Coste de licencias software

En este apartado se calculan los costes asociados al software utilizado durante el desarrollo del proyecto. Tanto el utilizado para su diseño e implementación, como el empleado en la tarea de documentación.

| Software | Nº de licencias | Coste imputable(€)/Ud. | Total(€) |
|---|---|---|---|
| **SmartDraw 2012 Standard** | 1 | 162 | 162 |
| **Paquete Microsoft Office 2010 Profesional** | 1 | 499 | 499 |
| **Microsoft Visio 2010 Standard** | 1 | 330 | 330 |
| **Microsoft Project 2010 Standard** | 1 | 775 | 750 |
| **IDE Eclipse Indigo** | 1 | 0 | 0 |
| **IDE Netbeans 7.0** | 1 | 0 | 0 |
| **Java Development Kit 6** | 1 | 0 | 0 |
| | | | **1741 €** |

Tabla 3: Coste de licencias de software

## Coste de hardware

Es necesario también calcular el coste que requeriría comprar los componentes hardware; imprescindibles para llevar a cabo el desarrollo de un proyecto informático

como éste. Por lo tanto este tipo de costes, calculados mediante la fórmula de amortización, también se detallan a continuación.

| Hardware | Cantidad | Coste(€) /Ud. | % de uso | Dedicación (meses) | Periodo de depreciación | Total(€) |
|---|---|---|---|---|---|---|
| **Ordenador Portátil Dell Inspiron 6400** | 1 | 335 | 100 | 12 | 60 | 67 |
| **Ratón Logitech** | 1 | 56 | 100 | 12 | 60 | 11,20 |
| **Impresora HP LaserJet 1010** | 1 | 110 | 100 | 12 | 60 | 22 |
| | | | | | | **100,20 €** |

Tabla 4: Coste del hardware

La fórmula utilizada para el cálculo de la amortización es:

$$\frac{A}{B} xCxD$$

Donde: A = nº de meses desde la fecha de facturación en que el equipo es utilizado.
B = periodo de depreciación (60 meses).
C = coste del equipo.
D = % del uso que se dedica al proyecto (habitualmente 100%).

## Otros costes del proyecto

En la siguiente tabla se describen todos los costes del proyecto que no han sido contemplados en apartados anteriores.

| Descripción | Total(€) |
|---|---|
| **Material de oficina** | 35 |
| **Recambios de impresora HP LaserJet 1010** | 97 |
| | **132 €** |

Tabla 5: Otros costes

## Resumen de costes

En este último apartado se muestra un resumen de todos los costes detallados anteriormente y se calcula el total, al que se le aplica una tasa de costes indirectos del 20%.

| Concepto | Total(€) |
|---|---|
| **Personal** | 27810 |
| **Licencias software** | 1741 |
| **Amortización hardware** | 100,20 |
| **Otros costes** | 132 |
| | **29783,2€** |

Tabla 6: Resumen de costes

Para calcular el total le aplicamos una tasa de costes indirectos del 20% que determinará el presupuesto final. Por lo tanto:

"El presupuesto total de este proyecto asciende a la cantidad de **TREINTA Y CINCO MIL SETECIENTOS TREINTA Y NUEVE** EUROS.

Leganés a 10 de Julio de 2012

El ingeniero proyectista

Fdo. Bruno Martín Gayral

# Anexo B.

# Manual de usuario

En este documento se explican los pasos a seguir para ejecutar el programa y llevar a cabo partidas contra Uc3mBot mediante la interfaz gráfica.

La aplicación se compone de 3 programas:

- El servidor de juego: Encargado de dirigir la partida, informando a todos los jugadores de cómo se desarrolla cada mano, así como de sus cartas y otra información relevante.
- El interfaz de juego: Utilizado para poder realizar partidas contra Uc3mBot.
- El jugador Uc3mBot.

## Requisitos previos

Tanto Uc3mBot como el servidor y la interfaz de juego han sido desarrollados en Java sobre un entorno Linux, concretamente la distribución Ubuntu 11.10. El servidor y la interfaz deben poder ser ejecutados en otros entornos que tengan Java instalado en su versión 6 de Oracle. Sin embargo Uc3mBot incorpora librerías de código en C que han sido desarrolladas específicamente para entornos Linux y por lo tanto sólo puede ejecutarse en dichos entornos. En caso de no tener instalada la versión 6 de Java Oracle, ésta puede obtenerse en java.sun.com.

Una vez configurado el entorno, se debe descomprimir el archivo Uc3mBot.tar.gz, de forma que se creará una carpeta llamada “Uc3mBot” con los siguientes elementos:

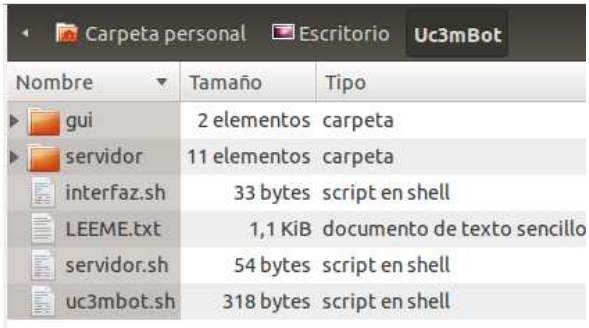



Figura 17: Archivos dentro de la carpeta Uc3mBot

# Ejecución de la aplicación

### 1. *Iniciar el servidor de juego*

Abrir una consola y situarse dentro de la carpeta descomprimida "Uc3mBot".

Ejecutar entonces el servidor de juego mediante el comando "./servidor.sh" tal y como

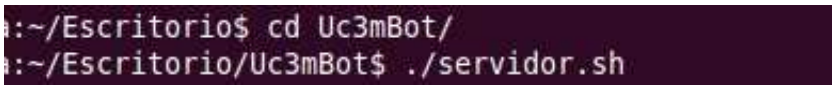



Figura 18: Ejecución del servidor de juego

### 2. *Iniciar el agente Uc3mBot*

Abrir una segunda consola y situarse igualmente dentro de la carpeta descomprimida "Uc3mBot".

A continuación ejecutar el programa Uc3mBot a través del comando "./uc3mbot.sh"

### 3. *Iniciar la interfaz de juego*

Abrir una tercera consola y situarse dentro de la carpeta descomprimida "Uc3mBot".

Ejecutar el comando "./interfaz.sh"

### 4. *Conectar la interfaz de juego con el servidor*

Una vez abierta la ventana de la interfaz de juego, desplegar la lista de partidas activas en el servidor y hacer doble clic sobre la partida que aparece nombrada como "ROOM_1" (Véase **¡Error! No se encuentra el origen de la referencia.**). El servidor nterpretará dicha orden y procederá a iniciar la partida.


Figura 19: Conexión de la interfaz de juego con el servidor

## Controles del juego

La interfaz consta de la mesa de juego, donde se muestra toda la información necesaria sobre el transcurso de cada una de las manos jugadas. Debajo del nombre de cada jugador se encuentra la cantidad de dinero o fichas por la que se está jugando la mano, dicha cantidad es reiniciada al principio de cada mano, pues son los marcadores situados en la parte izquierda de la ventana los que indican el total de ganancias (en verde) o pérdidas (en rojo) acumuladas. También se muestra en todo momento el tamaño del bote por el que se está jugando (Véase **¡Error! No se encuentra el origen de la referencia.**).

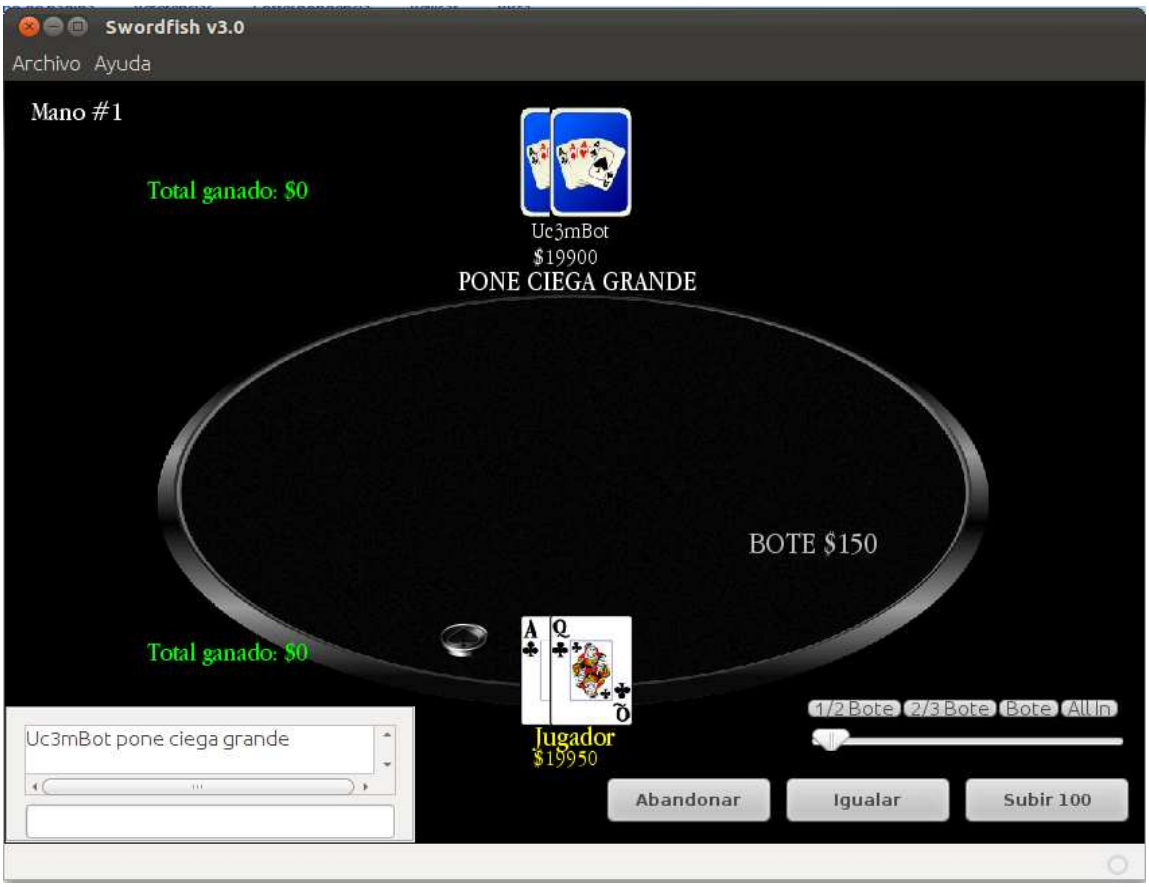



Figura 20: Interfaz de juego

Para poder realizar las acciones de juego se han incluido tres botones en la esquina inferior derecha de la ventana que realizan una acción determinada cada uno, estas son:

- Abandonar (tecla de acceso rápido ALT+F)
- Igualar o Pasar (tecla de acceso rápido ALT+C)
- Subir una cantidad X (tecla de acceso rápido ALT+R)

Sobre estos tres botones principales se encuentran otra serie de botones de menor tamaño y una barra deslizadora cuya función es ayudar al usuario a escoger el tamaño de la apuesta. Concretamente son:

- 1/2 Bote: Para apostar la mitad del bote actual
- 2/3 Bote: Para apostar dos tercios del bote actual

- Bote: Para hacer una apuesta igual a la cantidad de dinero acumulado en el bote
- All in: Para apostar todas las fichas que nos quedan.


Figura 21: Botones y barra deslizadora para seleccionar el tamaño de la apuesta

Tras escoger el tamaño de la apuesta, se debe pulsar el botón subir para hacer efectiva la acción.

En la esquina inferior izquierda de la ventana de juego se sitúa la consola. En ella se irán escribiendo automáticamente todas las acciones que han ido ocurriendo durante el transcurso de la partida. También se puede escribir texto en ella y comunicarse con el rival, especialmente útil a la hora de jugar contra otro humano (Uc3mBot no está diseñado para responder mensajes).

Se puede salir de la partida en cualquier momento cerrando la ventana, pero si ya se ha realizado al menos una acción en el transcurso de la mano actual, el dinero apostado se quedará en el bote.

# Anexo C.

# Manual de referencia

En el presente documento se describen los detalles relacionados con la implementación, así como las herramientas y programas utilizados durante esta fase del proyecto. Con esta información se pretende facilitar posibles ampliaciones o modificaciones futuras de la aplicación.

## Entorno de desarrollo

El programa ha sido escrito en el lenguaje de programación Java y compilado y ejecutado mediante la versión 1.6.0_26 de Oracle. Se utilizó para su desarrollo el entorno eclipse en su versión Indigo para Java, que puede descargarse desde la página de Eclipse en www.eclipse.org/downloads.

## Clases y librerías utilizadas

El proyecto "Uc3mBot2" (denominado así porque es la segunda versión de Uc3mBot) consta de las clases mostradas en la Figura 22.

La mayoría del desarrollo se ha realizado en el lenguaje de programación Java, sin embargo el programa utiliza la librería "XPokerEquity" (desarrollada por James Devlin y descargable gratuitamente desde www.codingthewheel.com/file.axd?file=XPokerEquity.zip) cuyo código se ha adaptado para poder ejecutarse en un entorno Linux. Ésta librería es la encargada de realizar gran parte de los cálculos de valor esperado, y está desarrollada C, puesto que este lenguaje permite una mayor velocidad de cálculo.

Los interfaces desarrollados para comunicar la parte de código escrita en Java con la librería XPokerEquity están escritos en el lenguaje Java Native Interface (JNI), que es un infraestructura digital de programación que permite que un programa escrito en Java ejecutado en la máquina virtual java (JVM) pueda interactuar con programas escritos

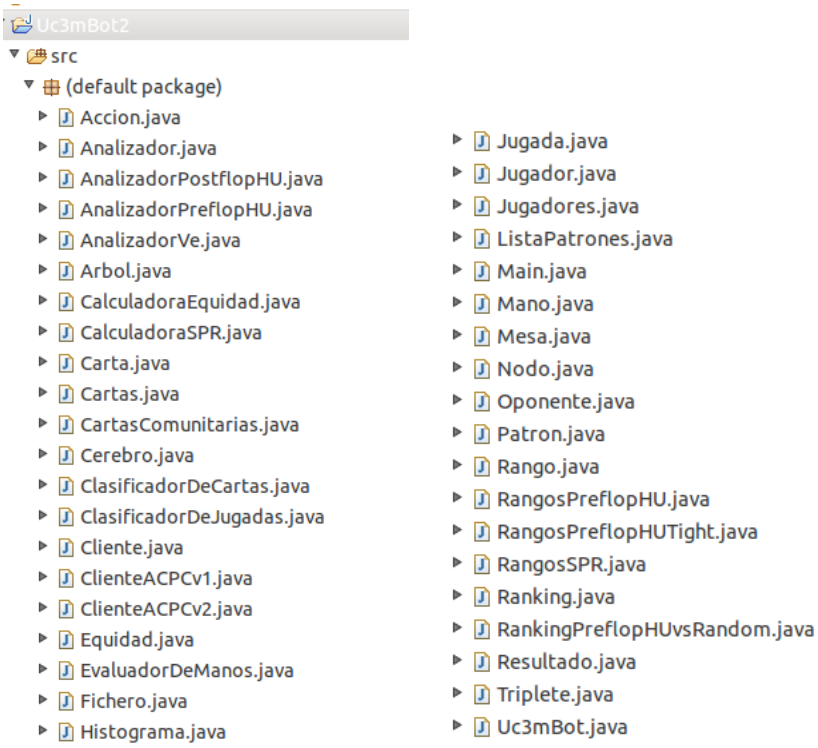



Figura 22: Clases de que consta el proyecto Uc3mBot

en otros lenguajes como C, C++ y ensamblador. Más información acerca de cómo programar en éste lenguaje puede encontrarse en (24) también descargable en internet desde java.sun.com/docs/books/jni/download/jni.pdf.

El interfaz escrito en lenguaje JNI está compuesto de los documentos "CalculadoraEquidad.c" y "CalculadoraEquidad.h".

Sin embargo no es necesario modificar dichos archivos ya que se han creado de manera que permitan acceder a todas las rutinas necesarias de dichas librerías desde la clase "CalculadoraEquidad.java" del proyecto Uc3mBot. La compilación de estos archivos da como resultado la librería "libpokerjni.so" que ha de estar situada en la misma carpeta en que esté el ejecutable del proyecto "Uc3mBot2.jar".

Además de la librería antes mencionada, se ha utilizado una versión en Java de la librería poker-eval, encargada ésta de evaluar el valor de cada mano de póquer. Para poder utilizarla se debe importar el archivo "poker-eval.jar" al proyecto. Ésta librería fue desarrollada a través del proyecto "pokersource" de código abierto, del que se puede encontrar más información en gna.org/projects/pokersource/. Sin embargo el paso de incluir dicha librería no es necesario ya que viene incluida en el archivo comprimido del proyecto. La clase que utiliza los métodos de ésta librería es "EvaluadorDeManos.java".

## Observaciones importantes

Puesto que Uc3mBot se conecta al servidor mediante un script, se ha redirigido la salida estándar al fichero executionLog.txt en la carpeta /bots del servidor, para así poder observar los mensajes que escribe durante su ejecución.

Durante la ejecución también se genera un archivo denominado historial.txt donde se imprimen los resultados de cada partida medidos en ciegas grandes. Es decir, si en la última mano Uc3mBot perdió 1600$ y la ciega grande vale 100$, en el historial se creará una línea con el valor -16, porque en esa mano se perdieron 16 ciegas grandes.

## Instrucciones para futuras modificaciones del código

Para realizar alguna modificación del código lo más aconsejable es instalar el entorno de desarrollo Eclipse indicado más arriba, e importar el proyecto directamente desde el archivo comprimido "Uc3mBot v1.1.zip". En el menú File->Import se debe seleccionar la opción "ExistingProjects into Workspace" tal y como muestra la **¡Error! No se encuentra el origen de la referencia.**.

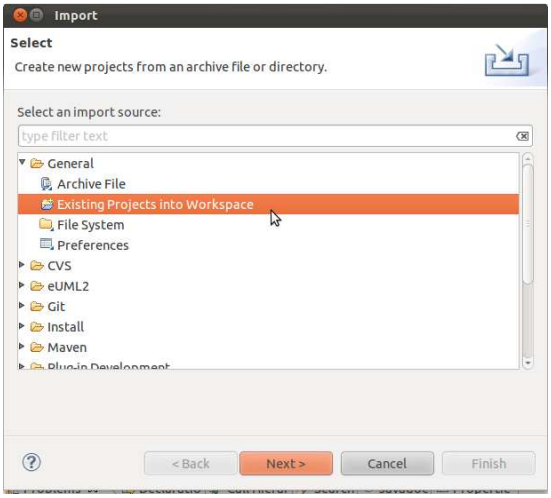



Figura 23: Importación del proyecto Uc3mBot al entorno de desarrollo Eclipse

Después en la ventana siguiente se hace clic sobre "Select archive file" y "Browse", entonces se elige el archivo .Zip del proyecto, finalmente se hace clic sobre "Finish" y tendremos el proyecto importado.

## Instrucciones para la compilación

El proyecto Uc3mBot debe compilarse en un archivo .jar ejecutable para poder utilizarlo posteriormente como jugador. Dicho ejecutable se puede generar directamente desde Eclipse, seleccionando el proyecto y haciendo clic en el menú File->Export, después se selecciona "Runnable JAR File" tal y como se muestra en la **¡Error! No se encuentra el origen de la referencia.**, y se hace clic en Next.

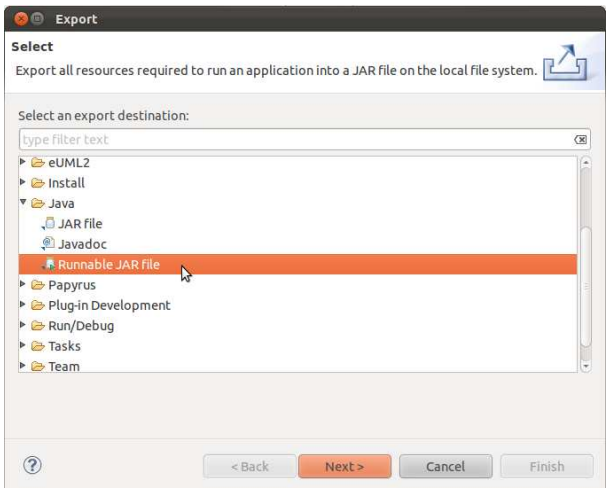



Figura 24: Generación del archivo ejecutable .jar

Entre las opciones de creación del ejecutable se debe escoger “Extract required libraries into generated JAR” tal y como muestra la **¡Error! No se encuentra el origen de la referencia.**. Finalmente hacer clic en “Finish” para terminar el proceso.

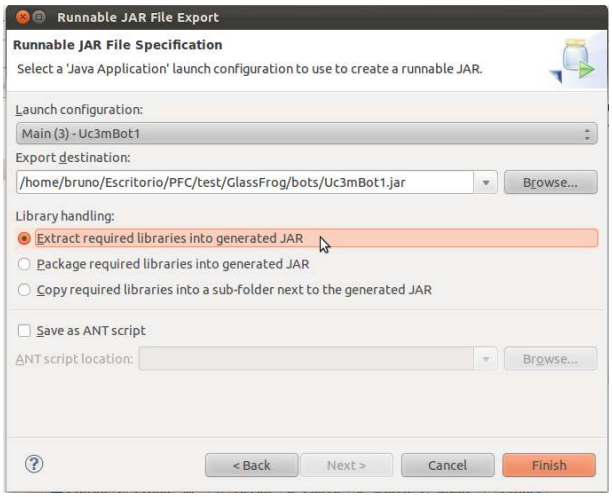



Figura 25: Opciones para generar el ejecutable .jar

## Detalles del servidor y la interfaz gráfica

GlassFrog es el servidor de póquer utilizado, una distribución de la Annual Computer Poker Competition (ACPC), que se realiza cada año en la Conferencia Internacional sobre Inteligencia Artificial AAAI y que atrae competidores de todo el mundo. Dicha competición desarrolló su propio servidor de juego de forma que todos los participantes siguiesen unos mismos estándares de juego. La versión del protocolo del servidor utilizada en este proyecto es la 1.0.0, que se comunica con el cliente de la clase ClienteACPCv1. Más recientemente la ACPC publicó un nuevo servidor que implementaba un protocolo parecido, cuya versión es la 2.0.0, que es la que actualmente se utiliza en dicha competición.

En este proyecto se utiliza la primera versión por defecto, ya que la versión más reciente no es compatible con la interfaz gráfica. Un ejemplo del protocolo 1.0.0 utilizado se encuentra en http://webdocs.cs.ualberta.ca/~pokert/code/nolimitprotocol.pdf.

Sin embargo también se puede utilizar el programa con los últimos servidores de juego publicados por la ACPC, para ello simplemente se ha de cambiar el tipo de cliente creado desde la clase Main a un objeto de la clase ClienteACPCv2.

La interfaz gráfica es una versión del proyecto Swordfish, proporcionado también por la ACPC. Ésta interfaz fue desarrollada para utilizarse junto a la versión 1.0.0 del servidor GlassFrog y para el motivo del presente proyecto se le han realizado una serie de modificaciones, como la traducción al español, y una muestra de la información de juego más intuitiva. Dichas modificaciones sobre la interfaz se han llevado a cabo con el entorno de desarrollo NetBeans 7.0.1 disponible desde la página netbeans.org/downloads/.

# Anexo D.

## Diagrama de Clases Completo

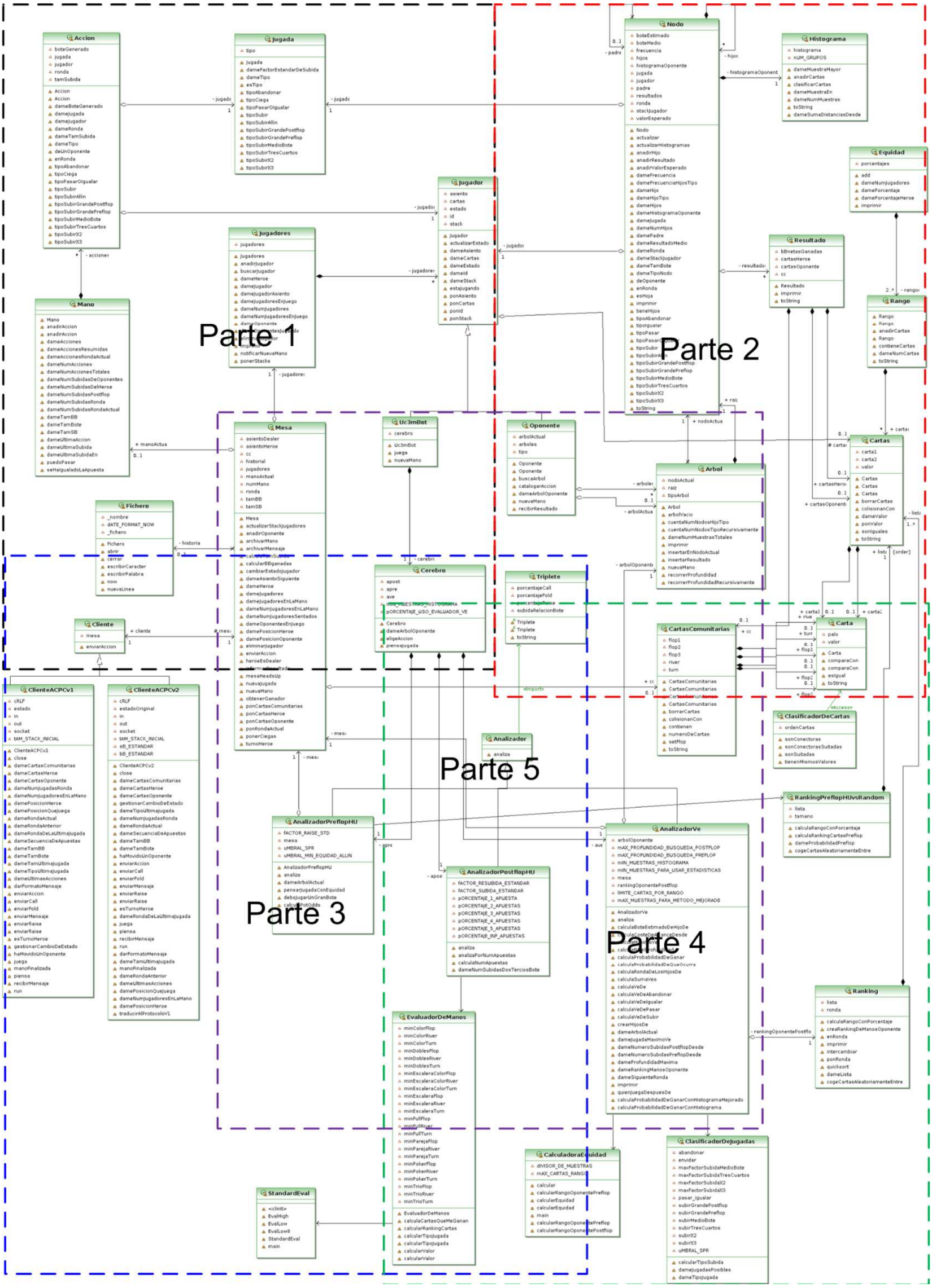



Figura 26: Diagrama de clases fragmentado

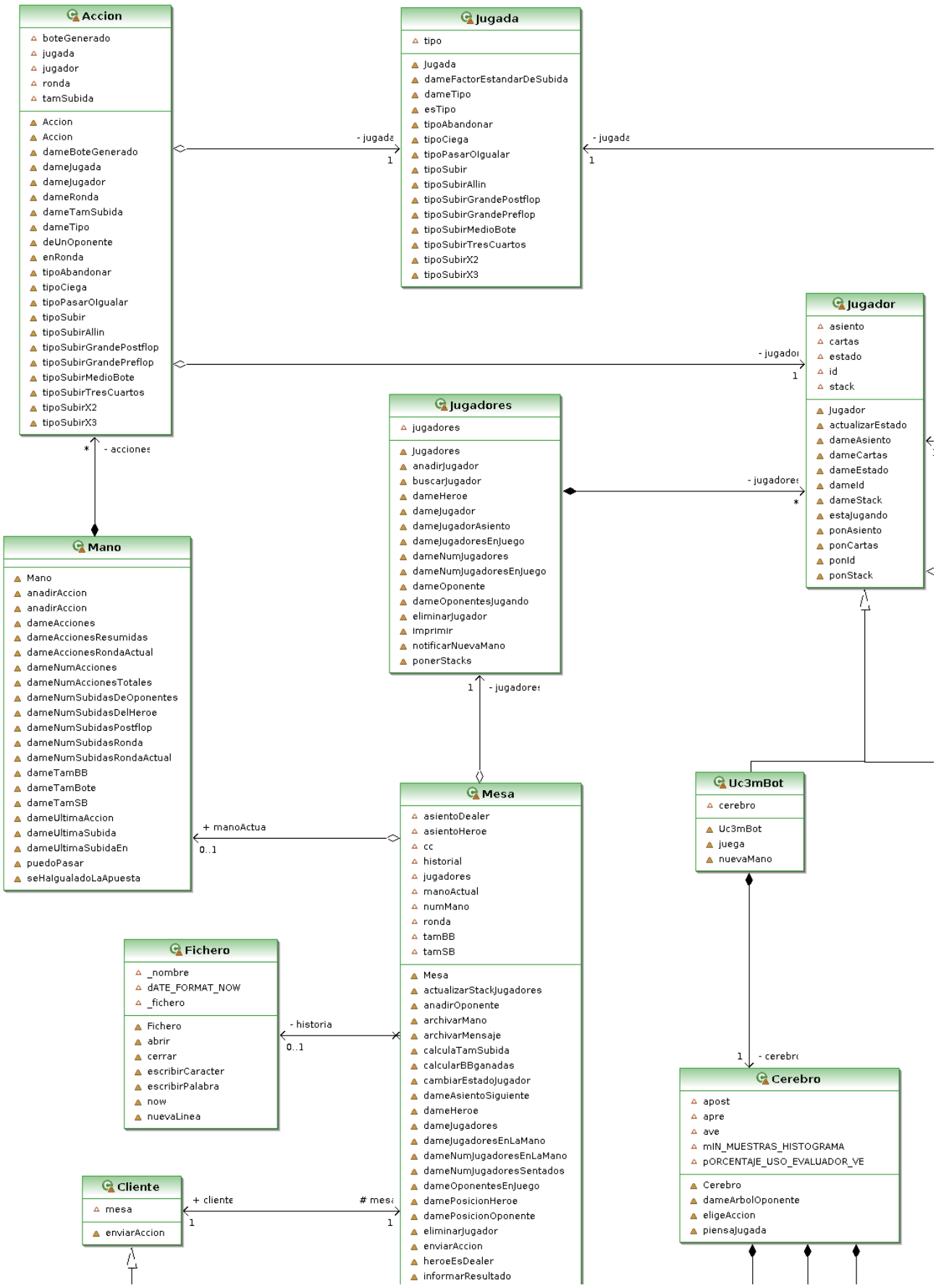



Figura 27: Diagrama de clases (parte 1)

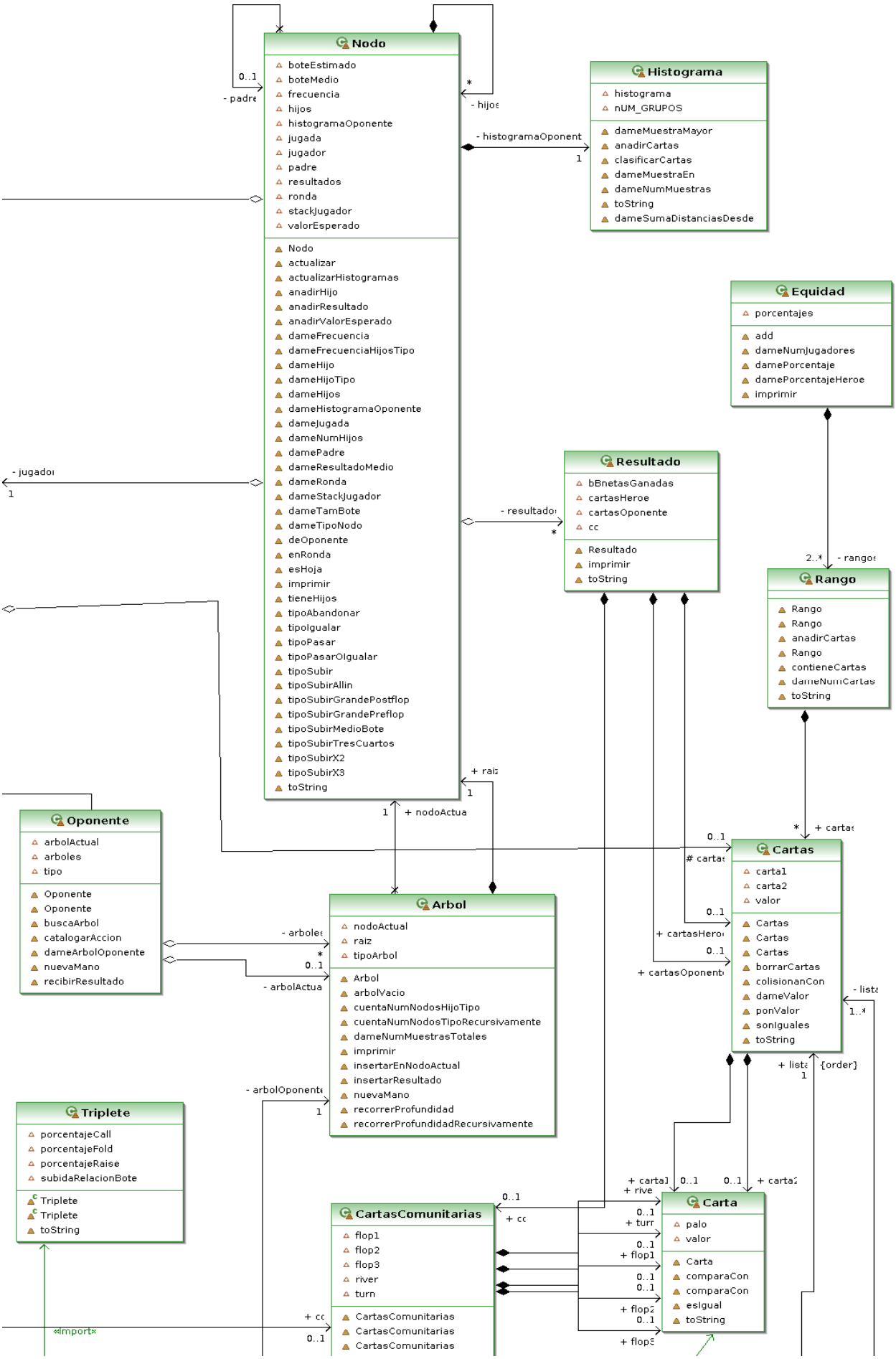



Figura 28: Diagrama de clases (parte 2)

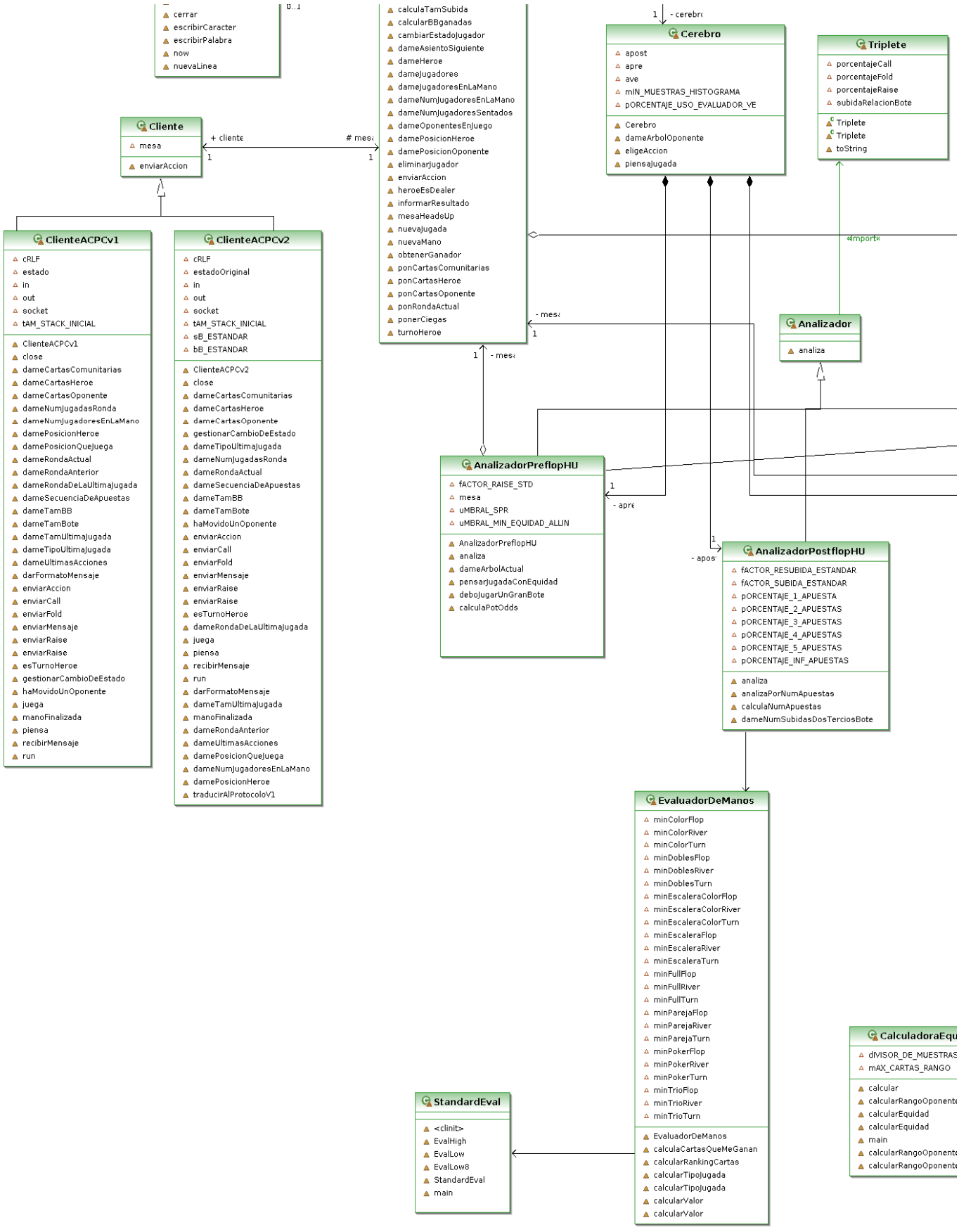



Figura 29: Diagrama de clases (parte 3)

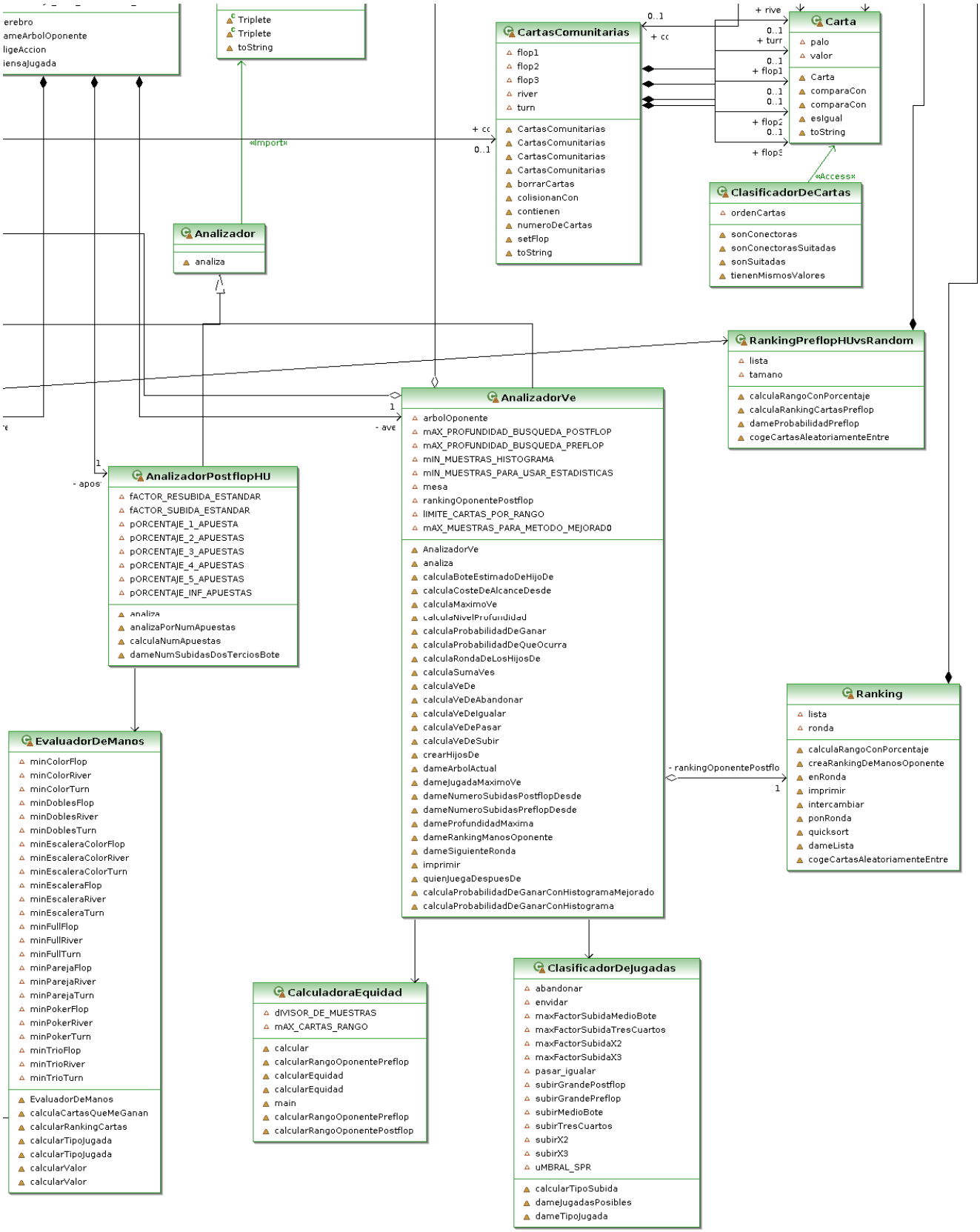



Figura 30: Diagrama de clases (parte 4)

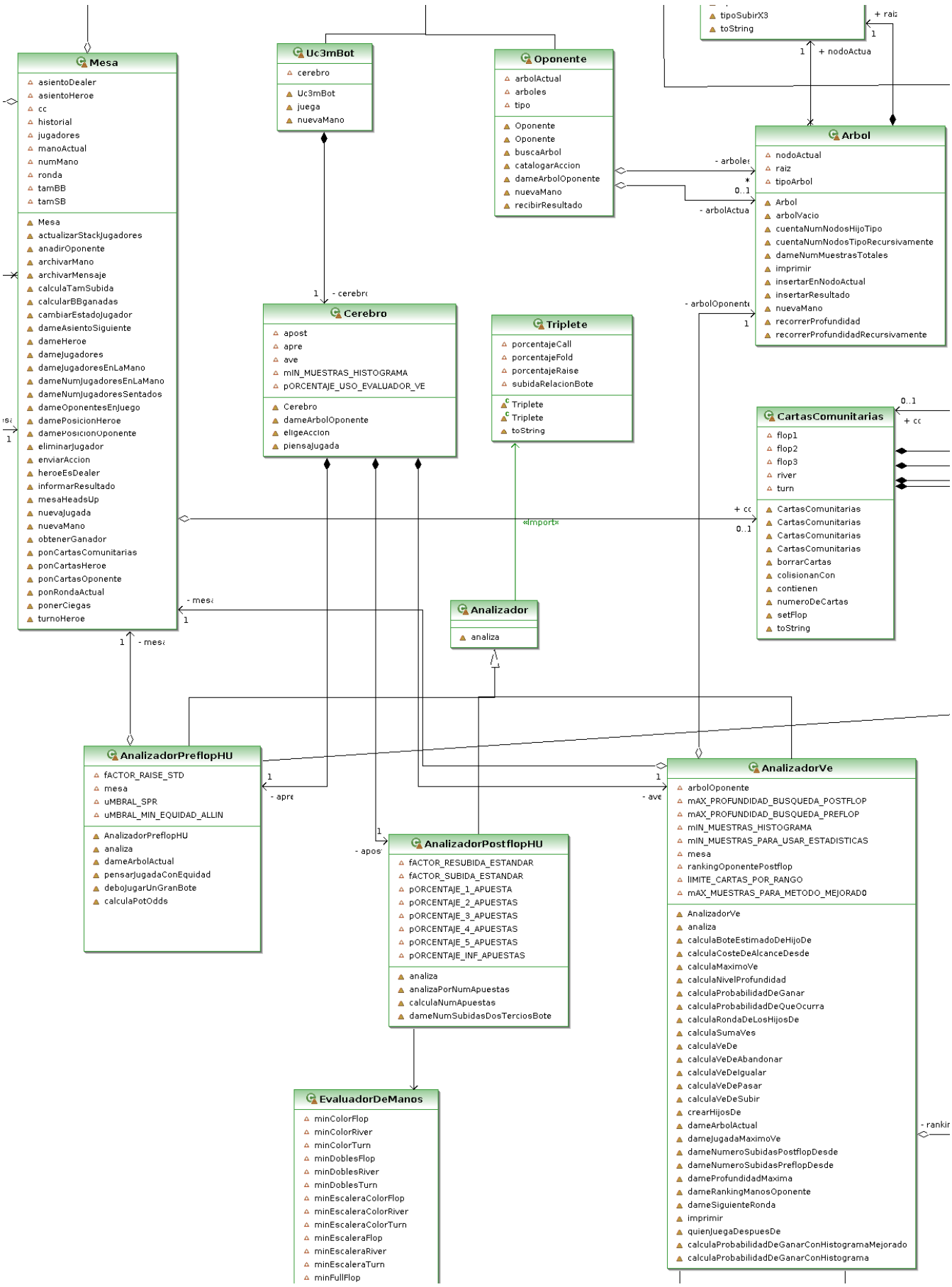



Figura 31: Diagrama de clases (parte 5)

# Anexo E.

# Reglas del Texas Hold'em

El póquer Texas Hold'em es un juego de cartas para 2 o más jugadores cuyo objetivo es maximizar las ganancias. Actualmente es la variante de póquer más popular y también es la más jugada en el mundo. Su versión sin límite de apuestas (no limit) es la utilizada en los eventos principales de series mundiales de póquer y los torneos más importantes. Cuando se juega entre dos jugadores se le denomina "Heads-up", entre tres y seis jugadores se le denomina "Mesa Corta" y con siete jugadores o más se le denomina "Mesa Larga".

## Cómo jugar

Hold'em se suele jugar usando las apuestas ciega pequeña ("small blind") y ciega grande ("big blind"). Estas apuestas se denominan "ciegas" ya que el jugador apuesta sin haber visto ninguna de las cartas de la mesa.

La ciega pequeña la pone el jugador a la izquierda del repartidor, y equivale a la mitad de la ciega grande.

La ciega grande la aporta el jugador a la izquierda del anterior y equivale a la apuesta mínima. En los torneos, la cantidad de las ciegas va aumentando conforme el torneo avanza.

Un botón (o "dealer") se utiliza para representar al jugador situado en la posición del repartidor. El botón de repartidor gira en la dirección de las agujas del reloj después de cada mano, con lo que se van turnando el repartidor y las "ciegas".

## Rondas de juego

### *1ª Fase, Pre-Flop*

El juego comienza repartiendo 2 cartas boca abajo a cada jugador. Estas cartas se denominan cartas privadas o “hole cards”. Son las únicas cartas que cada jugador recibirá individualmente, y serán descubiertas al final de la mano o si todos los jugadores apuestan "all-in".

La mano comienza con una ronda de apuestas antes de mostrar cartas comunitarias, comenzando por el jugador a la izquierda de la ciega grande y se continúa en la dirección de las agujas del reloj hasta que todos los jugadores igualan las apuestas.

Después de esta ronda de apuestas, si aún continúan al menos 2 jugadores en la mano, se comienzan a poner sobre la mesa las 5 cartas comunitarias descubiertas (de esta forma Texas hold'em es un juego de póquer "descubierto") en tres fases:

### *2ª Fase, Flop: Tres cartas descubiertas*

El repartidor saca el flop, tres cartas descubiertas simultáneamente y se realiza otra ronda de apuestas. Esta y las demás rondas de apuestas comienzan por el jugador a la izquierda del botón y continúan en el sentido de las agujas del reloj.

### *3ª Fase, Turn: La cuarta carta es descubierta*

Después de la ronda de apuestas en el "flop", una nueva carta descubierta “turn” se pone sobre la mesa, seguido de otra ronda de apuestas. A esta carta también se la denomina 4ª calle ("fourth street")

### *4ª Fase, River: Se descubre la última carta*

Después del "turn", otra carta descubierta “river” se coloca sobre la mesa (formando las 5 cartas comunitarias) y es seguido de otra ronda de apuestas. A esta carta también se la denomina 5ª calle ("fifth street").

### *Se determina el ganador*

Si un jugador apuesta y todos los demás se retiran, el jugador se lleva todo el dinero o fichas del bote, y no tendrá que mostrar sus cartas. Si dos o más jugadores continúan

después de la última ronda de apuestas, se muestran las cartas (showdown). En la muestra, cada jugador utiliza las cinco mejores cartas posibles de entre las siete que forman sus dos cartas privadas y las cinco cartas comunitarias de la mesa.

## Ranking de jugadas

| | Jugada | Descripción | Ejemplo |
|---|---|---|---|
| 1 | **Escalera real** | Cinco cartas seguidas del mismo palo del 10 al As. |  |
| 2 | **Escalera de color** | Cinco cartas consecutivas del mismo palo. En caso de empate decide la carta más alta. |  |
| 3 | **Póquer** | Cuatro cartas iguales en su valor. En caso de que varios jugadores tengan póquer, gana el que tenga el póquer con las cartas más altas. Si varios jugadores tienen el mismo póquer (si éste se da en las cartas comunitarias) gana el que tenga la carta más alta. |  |
| 4 | **Full** | Tres cartas iguales (trío), más otras dos iguales (par). En caso de que varios jugadores tengan full, gana el que tenga el trío más alto y luego el que tenga la pareja más alta. |  |
| 5 | **Color** | Cinco cartas del mismo palo, Gana la carta más alta dentro del mismo palo. |  |
| 6 | **Escalera** | Cinco cartas consecutivas de palos diferentes. Gana la escalera más alta. |  |
| 7 | **Trío** | Tres cartas iguales en su valor. En caso de empate decide el trío con las cartas más altas y en caso de nuevo empate decide la carta o cartas más altas de las que no forman trío. |  |
| 8 | **Dobles parejas** | Dos pares de cartas. En caso de empate decide la pareja más alta y si continúa el empate decide la segunda pareja más alta y en caso de nuevo empate decide la carta suelta más alta. |  |
| 9 | **Pareja** | Dos cartas iguales y tres diferentes. En caso de empate decide la pareja más alta y si continúa el empate deciden las cartas más altas entre las restantes. |  |
| 10 | **Carta alta** | Gana quien tiene la carta más alta de todas. Si hay empate con la primera carta, se continúa con las siguientes para decidir quién tiene la carta más alta. |  |

Si dos jugadores o más comparten la mejor mano, entonces el bote se divide por igual entre los jugadores.

## Modalidades según las apuestas

### *Límite fijo (Fixed limit)*

La modalidad con límite fijo (o simplemente "con límite") es una modalidad en la que sólo se puede apostar una cantidad de fichas fija.

Las apuestas en las 2 primeras rondas (pre-flop y flop) son iguales a la ciega grande, y en las 2 siguientes rondas (turn y river) las apuestas son iguales al doble de la ciega grande.

### *Sin límites (No limit)*

El Texas Hold'em sin límite de apuesta es una modalidad en la que es posible apostar cualquier cantidad de fichas que se tenga sobre la mesa por encima de la apuesta mínima. Apostar todas las fichas esto se conoce como "All in" (Voy con todo).

Es la modalidad más conocida de este juego y hay que tener grandes conocimientos para jugarla, ya que se pueden ganar o perder botes mucho más grandes más fácilmente.

### *Limitado al bote (Pot limit)*

Como máximo puedes apostar la cantidad de fichas que se encuentre en el bote en el momento de tu apuesta. Como bote se entiende la suma del bote y todas las apuestas de la mesa más la cantidad que el jugador debe igualar antes de subir.

Es la variante menos común.

# Anexo F.

## Terminología de los juegos de Póquer

- Absolute nuts: Jugada más alta que se puede formar, y que no puede ser superada por ninguna otra combinación.
- Add-On: Oportunidad final para añadir fichas en un torneo que admita recompras. Sólo se puede hacer una vez y cierra el periodo de recompras.
- All in o envidarse: Acción de juego. Estamos all in cuando llevamos todas nuestras fichas al centro de la mesa en una partida.
- Backdoor: Proyecto que necesita que las cartas del turn y el river sean favorables.
- Bad beat: Un golpe de mala suerte. Se suele denominar un Bad Beat a cuando pierdes una mano que creías tener ganada por una jugada absurda de un oponente.
- Barrigona o "gutshot": Escalera a falta de una carta interior, o sea con tan sólo 4 outs. Por ejemplo flop 89A, con 56 en mano. Un 7 nos da la escalera.
- Apuesta grande o "big bet": En los juegos con límite de apuestas son las apuestas que se realizan en las dos últimas rondas del juego, frente a las "small bets" de las dos primeras.
- Botón: Repartidor o "dealer". El jugador que tiene la última posición de la mesa y por tanto es último en hablar.
- Carta gratis: Conseguimos una carta gratis cuando, en una ronda de apuestas, no tenemos que hacer frente a ninguna apuesta para ver la siguiente carta.
- Check raise: Acción de juego. Consiste en pasar (check) el turno en una ronda de apuestas con la intención de subir (raise) si alguno de nuestros oponentes apuesta en la misma ronda.
- Pasar o "check": Ceder el turno de apuestas al siguiente oponente.

- Ciega grande o “big blind” (BB): Segunda posición a la izquierda del botón, y último jugador en apostar antes del flop. El jugador en esta posición ha de poner una cantidad establecida de antemano en el bote, antes de comenzar la mano.
- Ciega pequeña o “small blind” (SB). Posición en una mesa de hold'em tras el botón.
- Cold call: Se denomina cold call cuando un jugador iguala la apuesta ante un bote donde ha habido una apuesta y una subida. Es decir, ve una doble apuesta para seguir en la mano.
- Cut-off: Posición del jugador anterior al botón.
- Entrada o “buy in”: Cantidad de fichas que debes comprar para participar en un evento.
- Flop: Segunda ronda de apuestas, donde al comienzo de la misma se añaden tres cartas públicas al centro de la mesa, constituyendo el 'flop'.
- Fold: Acción de juego. Abandonar una mano, retirarnos. Podemos hacer fold cuando llegue nuestro turno y no queramos ver (hacer call) una apuesta, o sencillamente no queramos mostrar nuestra mano.
- Heads up: Hablamos de heads up cuando jugamos únicamente contra un oponente.
- Hijack: Posición en la mesa. El Hijack es el jugador inmediatamente a la derecha del "cutoff" (el que habla antes que él).
- Hold'em: Variante del póquer más jugada del mundo. El nombre completo es Texas Hold'em.
- Igualar la apuesta o “call”: Acción de juego. Hacemos call cuando habiendo apostado alguno de nuestros oponentes, igualamos su apuesta y cedemos el turno al siguiente.
- Ir de farol: apostar sin tener una mano ganadora con intención de que el adversario abandone
- Kicker: Carta más alta en la que apoyar nuestra jugada. Ejemplo: Si el flop es AJ7, y nuestras cartas privadas son AK. Tenemos pareja de ases, con “kicker” K.
- Limit: Modalidad del póquer en la que la cantidad máxima a apostar en cada ronda de apuestas viene determinado por el límite de la mesa. Por ejemplo, en

una mesa limit de ciegas 1/2, la apuesta máxima es 1 en las dos primeras rondas de apuestas, y 2 en las siguientes.

- Limp: Entrar al bote sin subir, igualando.
- Loose: Jugador que juega un rango amplio de cartas.
- Multicolor: Se habla de un flop multicolor, cuando no hay dos cartas del mismo palo, de forma que no se puedan formar colores.
- No limit: Modalidad de póquer en la que no hay límite máximo a la cantidad que puedes apostar en cada ronda de apuestas.
- Nuts: Hablamos de las "nuts" cuando tenemos la jugada más alta posible que se puede formar con las cartas de la mesa.
- Odds: Forma de expresar las posibilidades de que algo ocurra frente a que no ocurra. Por ejemplo, la probabilidad de que salga un seis en el lanzamiento de un dado es un 16% (1/6*100). Esta misma probabilidad expresada en "odds" sería 1:5 o 1 a 5. De cada 6 lanzamientos, 1 saldrá un seis, mientras en los otros 5 no. Podemos expresar las "odds" inversamente, como la probabilidad de que algo no ocurra frente a que ocurra. En el ejemplo anterior 5:1 ó 5 a 1. Esta forma es la utilizada comúnmente en el póquer.
- Offsuited: Ver suited. Dos cartas o más son están "offsuited" cuando no comparten el mismo palo.
- Outs: Término relacionado con la probabilidad. Un "out" es cada una de las posibles cartas que completan nuestra jugada. Por ejemplo, con una pareja en mano, tenemos 2 outs para completar el trío.
- Overcards: Cartas altas. Hablamos de "overcards" cuando nuestras cartas privadas son más altas que las que constituyen las cartas de la mesa. Por ejemplo, AK con flop 59J, son "overcards".
- Overpair: Tenemos una "overpair" cuando tenemos una pareja en mano que es más alta que cualquier pareja formada con las cartas de la mesa. Por ejemplo, teniendo KK, con un flop 79Q.
- Pot limit: Variante del póquer Texas Hold'em parecida al sin límite. En este formato, la cantidad máxima que puedes apostar en una ronda de apuestas viene determinado por el tamaño del bote.
- Preflop: Antes del flop. Normalmente nos referiremos a lo que haya pasado antes de ver el flop.

- Proyecto: Cuando estamos a la espera de las siguientes cartas para completar nuestra mano. Por ejemplo, estamos esperando obtener un color, o a una escalera.
- Rake: Porcentaje de cada bote que la sala de póquer aparta como comisión por sus servicios.
- Razz: Variante de póquer. Seven Card Stud jugado por lo bajo, es decir, la mejor mano es A2345.
- Recompra: En algunos torneos es posible comprar más fichas cuando te eliminan. Es lo que se denominan como “rebuys” o recompras.
- Resubir: Consiste en volver a subir la apuesta, una vez que alguno de nuestros oponentes ya ha subido en la misma ronda.
- River: Última carta comunitaria y cuarta y definitiva ronda de apuestas en una partida de póquer Texas Hold'em.
- Semi farol: Jugada técnica que consiste en intentar un farol con una jugada, que sin tener ningún valor en el momento, puede ganar con la siguiente carta.
- Showdown: Momento en el que los jugadores enseñan las cartas al finalizar una mano de póquer.
- Slowplay: Se hace “slowplay” cuando con una mano muy potente no se muestra fuerza para incitar a los oponentes a continuar en el bote e intentar obtener mayores ganancias.
- Subir: Subir la apuesta anterior apostando una cantidad mayor. De esa forma los oponentes que aún no hayan actuado tienen que hacer frente a dos apuestas para continuar en la mano.
- Suited: Hablamos de un flop o cartas “suited” cuando comparten el mismo palo. Por ejemplo, si en nuestra mano tenemos 78 de corazones, tenemos 78 suited.
- Tight: Denominamos tight a aquellos jugadores que siguen un estilo de juego selectivo. Un jugador tight sólo jugará las mejores cartas que pasen por sus manos.
- Top pair: Pareja más alta. Hablamos de top pair cuando tenemos la pareja más alta de las que se pueden formar en el flop.
- Turn: Tercera ronda de apuestas en una partida de póquer Texas Hold'em.

- UTG: Abreviatura proveniente del término inglés "Under The Gun". Se utiliza para referirnos al primer jugador en hablar preflop, es decir, el jugador a la izquierda de la ciega grande.